Asistente móvil para la interpretación de texto dirigido a personas con discapacidad cognitiva

Trabajo de Fin de Grado
Curso 2018–2019

Autores
Elianni María Agüero Selva
Ignacio Sande Soltero

Directoras
Raquel Hervás Ballesteros
Susana Bautista Blasco

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Asistente móvil para la interpretación de texto dirigido a personas con discapacidad cognitiva

Trabajo de Fin de Grado en Ingeniería Informática
Departamento de Ingeniería de Software e Inteligencia Artificial

Autores
Elianni María Agüero Selva
Ignacio Sande Soltero

Directoras
Raquel Hervás Ballesteros
Susana Bautista Blasco

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

31 de mayo de 2019

Agradecimientos

No podemos empezar a escribir este trabajo sin antes mostrar nuestro agradecimiento a nuestras tutoras Raquel y Susi por toda la ayuda que nos han ofrecido durante el proyecto, por tanta paciencia y dedicación. Gracias por darnos la oportunidad de participar en un proyecto real y de permitir que este TFG no se quede archivado una vez finalizado.

También agradecer al colegio Estudio3 AFANIAS por dejarnos probar la aplicación. Gracias a Juanmi y a su clase por la buena acogida, por mostrar tanto entusiasmo y por sus agradecimientos. Gracias por permitirnos vivir una experiencia realmente única.

A mis padres, por todo el apoyo y la confianza que me han dado en todo momento. A mis amigos, por ayudarme siempre que lo he necesitado y por los buenos momentos. Por último, y especialmente, a Dani. Sin tu ayuda no hubiese sido capaz de avanzar tanto, gracias por tu paciencia infinita y por tus ánimos.

Elianni

A mi familia, por todo el apoyo y los ánimos que me han dado durante estos años de carrera. A mis amigos y compañeros sin los que esta experiencia no hubiese sido lo mismo.

Ignacio

A todos ellos, muchas gracias.

Resumen

Las discapacidades cognitivas son aquellas que pueden llegar a limitar a la persona que las sufre en sus habilidades para realizar tareas diarias, limitándola en cosas tan simples como puede ser la correcta comprensión de un texto. La pérdida de información que puede llegar a suponer no poder captar el correcto contexto de lo que estás leyendo o incluso directamente no entenderlo, puede hacer que esa persona tenga un proceso de aprendizaje mucho más lento o que incluso le impida desenvolverse en situaciones cotidianas.

Hoy en día contamos con una gran cantidad de servicios y tecnologías que permiten que este tipo de obstáculos se vean resueltos con mayor facilidad. Sin embargo, el medio en el que se encuentran o la forma de acceder a ellos no es totalmente viable, dependiendo de la situación y el contexto en el que se encuentre la persona.

Por esta misma razón nace este proyecto, por la necesidad de acercar todos esos servicios al día a día de las personas con discapacidad cognitiva que tienen dificultades lectoras, y que en cualquier momento de necesidad tengan una herramienta de apoyo que puedan utilizar. La manera de conseguir esto es muy simple, es transformando el dispositivo móvil que siempre llevamos con nosotros en una herramienta de apoyo que de acceso a toda clase de servicios de ayuda de comprensión lectora. Este proyecto se basa en el desarrollo de una aplicación móvil, mediante la cual el usuario en caso de tener alguna dificultad a la hora de entender un texto cuente con las funcionalidades necesarias que le ayuden a comprenderlo. Las clases de funcionalidades con las que la aplicación contaría serían: transformación de texto a pictogramas, convertir una palabra compleja a pictogramas, obtener la definición de una palabra, etc.

Dada la importancia de trabajar directamente con los usuarios finales que van usar la aplicación, se ha seguido un diseño centrado en el usuario en colaboración con el Colegio Estudio3 AFANIAS, el cual, una vez se finalizo el desarrollo del proyecto nos permitió realizar una prueba de la aplicación con un grupo de alumnos del centro. Está evaluación con usuario finales nos permitió ver el impacto que tiene la aplicación en los usuarios para los que esta destinada.

Palabras clave

Accesibilidad, Aplicación móvil, Lectura Fácil, Pictogramas, Procesamiento de Lenguaje Natural, Diseño orientado al usuario, Servicios web.

Abstract

The cognitive disabilities are illnesses which can limit the abilities of people who suffer this kind of diseases, to the point of they could have some difficulties to make simple task like understand a text. The bad reading comprehension can cause that this people have a slower learning process or they can not cope with their daily situations.

Nowadays we can find a lot of services or technologies which allow that these types of barriers will be easily superated. But nevertheless the different means in which you can find these technologies or the way to access them is not universal and can prove a challenge depending on the living situation or circumstances of each individual.

For this reason we decided to do this project, for the necessity to make available this services to the people with cognitive disabilities for language comprehension, and can have this support tool in their phones for any time they need it. The way to get this objective it is easy, we transform the mobile which are always with us, in support tool that provide Access to these services to helping understand a text. This project is based on the development of a mobile application that will help its users with the understanding of text through the implementation different functionalities. The mobile application will count with functionalities such as conversion of text to pictograms, conversion of complex words to pictograms, show the different definitions of a specific word, etc.

Given the importance to work directly with the users who are going to use the mobile application, we have followed a user-centred design in collaboration with Colegio Estudio3 AFANIAS. When the project was finished, this school let us make a test of the application with their students. With this evaluation we could see the utility of the mobile application to people that was designed.

Key Words

Accessibility, mobile application, easy-reading, Pictograms, user-based design, web services.

Índice

Índice de figuras

Capítulo 1

Introducción

"Hemos venido a este mundo como hermanos;
caminemos, pues, dándonos la mano
y uno delante de otro"
— William Shakespeare

1.1. Motivación

Vivimos en un mundo donde cada vez es más frecuente el uso de la tecnología. En nuestro día a día vamos acompañados de dispositivos, los cuales de un modo u otro nos hacen las tareas más sencillas o nos permiten resolver los contratiempos que nos encontramos de una manera rápida y simple. Con esta idea en mente, nos planteamos qué podríamos hacer para acercar la tecnología a aquellas personas con discapacidad cognitiva que realmente tengan una necesidad real de contar con una herramienta de apoyo, que les facilitase su día a día.

Las personas con discapacidades cognitivas cuentan en ocasiones con una serie de limitaciones a la hora de comprender un texto, debido a la complejidad del lenguaje o a las expresiones cuyo contexto no logran entender. A causa de esta discapacidad determinadas tareas como leer un cartel informativo en el transporte público, leer un contrato de compra o estudiar una asignatura para un examen, pueden convertirse en una barrera que les dificulte continuar con su rutina a un ritmo normal o que les cause un proceso de aprendizaje más lento.

Cabe decir, que las discapacidades cognitivas en la comprensión de textos son generalmente permanentes, por lo que tiene un impacto importante en la vida de la persona y su familia.

De aquí surgió el crear una aplicación que les proporcionase a estas personas un modo de sobrepasar esas barreras que se puedan encontrar durante

el día. Además, qué mejor forma que mediante una aplicación móvil. Convirtiendo de esta manera, un elemento que todos llevamos con nosotros a todas partes en una herramienta verdaderamente útil.

1.2. Objetivo

El desarrollo de esta aplicación móvil consiste en la creación de una herramienta que ponga al servicio del usuario una serie de funcionalidades, las cuales le ayuden en la comprensión de un texto, frases o palabras complejas.

Nuestro principal objetivo es que dicha aplicación se convierta en una herramienta de referencia en aquellos momentos de dificultad. Es decir, en caso de necesitar un apoyo, la persona recuerde que en su teléfono cuenta con una aplicación la cual le va a simplificar las cosas.

La idea de que sea una aplicación móvil es poder llegar a la mayor cantidad de usuarios e intentar generalizar su uso. Además, se les proporciona a los usuarios un elemento que puedan incorporar en su día a día, ayudándoles así en su desarrollo autonómico.

En definitiva, una herramienta que permite al usuario final una mayor independencia y que facilite su proceso de autoaprendizaje.

1.3. Estructura del documento

Para una organización coherente de la memoria y de los pasos que hemos seguido para el desarrollo del proyecto, hemos dividido la memoria de la siguiente manera:

- **Capítulo 1**: en este capítulo, redactado en español e inglés, se comenta cual es la motivación y cuáles son los objetivos del trabajo. Además, se hace una estructura con un breve comentario de lo que se explicará en cada capítulo.
- **Capítulo 2**: para llevar a cabo la realización de este trabajo era importante informarnos sobre los tipos de discapacidad cognitiva que existen. Se exponen cuales son los Sistemas Aumentativos y Alternativos de la Comunicación (SAACs), centrándonos en el uso de los pictogramas y analizando algunas aplicaciones basadas en este tipo de SAAC. Además, se explica que es la Lectura Fácil y cuáles son las pautas a seguir para adaptar un texto a Lectura Fácil.
- **Capítulo 3**: en este capítulo hablamos sobre las tecnologías que hemos utilizado para desarrollar la aplicación. En nuestro caso hablamos de

Android, sistema operativo sobre el que se podrá utilizar la aplicación, y de Android Studio como entorno de desarrollo escogido para la implementación de la misma. Además, se exponen qué son los Servicios Web y que tipo utilizamos en el presente trabajo. El reconocimiento de texto es una parte importante de la herramienta y por tanto, nos hemos centrado en explicar cómo funciona y cuáles son los beneficios que aporta su uso. Por último, hemos hecho una pequeña explicación sobre el Procesamiento de Lenguaje Natural y hemos mencionado algunas tecnologías que existen actualmente.

- **Capítulo 4**: este capítulo se centra en el diseño de la aplicación. En esta parte hablamos de los distintos pasos que seguimos para alcanzar un diseño sencillo, intuitivo y finalmente amigable con el usuario.

- **Capítulo 5**: aquí se explica cómo está diseñada la arquitectura de la aplicación. Hacemos un recorrido por cada una de las pantallas explicando que funcionalidades se presentan en cada una de ellas. Además, explicamos algunos detalles internos de la implementación, es decir, cómo conectamos con los servicios, cómo procesamos el resultado, cómo almacenamos ciertos datos, etc.

- **Capítulo 6**: tras la implementación final de la aplicación, se llevó a cabo una prueba con usuario reales. En este capítulo se explica detalladamente cómo ha sido esta evaluación: el objetivo, el proceso y los resultados obtenidos.

- **Capítulo 7**: en español y en inglés, se presentan las conclusiones del presente trabajo, así como algunas ideas que han ido surgiendo a lo largo del desarrollo del mismo y las recomendaciones de los expertos durante la evaluación. Estas ideas y recomendaciones las hemos planteados como opción a desarrollar en el futuro.

- **Capítulo 8**: se exponen las aportaciones realizadas por cada uno de los miembros del proyecto.

Chapter 1

Introduction

1.1. Motivation

We live in a world where the use of technology is becoming more frequent. In our day to day we are accompanied by devices which, in one way or another, make our tasks easier or allow us to solve the setbacks that we find in a fast and simple way. With this idea in mind, we considered what we could do to bring technology closer to those people with cognitive disabilities who have a real need to have a support tool, to make their day to day easier.

People with cognitive disabilities sometimes have a number of limitations when it comes to understanding a text, due to the complexity of the language or the expressions whose context they do not understand. Because of this disability, certain tasks such as reading an informative poster on public transport, reading a contract of purchase or studying a subject for an exam, can become a barrier that makes it difficult for them to continue with their routine at a normal pace or that causes them a slower learning process.

It must be said that cognitive disabilities in the comprehension of texts are generally permanent, which is why it has an important impact on the life of the person and their family.

Based on this came the creation of an application that would provide these people with a way to overcome those barriers that may be encountered throughout their day to day life. Also, what better way than through a mobile application. Converting in this way, an element that we all carry with us everywhere in a truly useful tool.

1.2. Objective

The development of this mobile application consists in the creation of a tool that puts at the service of the user a series of functionalities, which will help them in the comprehension of a text, phrases or complex words.

Our main objective is that this application becomes a reference tool in those moments of difficulty. in other words, if you need support, remember that your phone has an application which will simplify things.

The idea for it to be a mobile application is to be able to reach as many users and try to generalize its use. Also, users are provided with an element that they can incorporate into their day-to-day life, helping them in their autonomous development.

In short, a tool that allows the end user greater independence and facilitates their learning.

1.3. Document structure

We have divided this assignment in eight chapters in order to achieve a coherent organization:

- **Chapter 1**: in this chapter we comment about the motivation and the objectives of the assignment. In addition, we made a brief explain of the structure of the each chapter.
- **Chapter 2**: for the realization of this project it was necessary to learn about the different types of cognitive disabilities. We explain about the Augmentative and Alternative Communication Systems (AAC) focusing on the use of the pictograms and analyzing some applications based on this type of AAC. In addition, it is explained that it is Easy Reading and what are the guidelines to follow to adapt a text to Easy Reading.
- **Chapter 3**: in this chapter we talk about the technologies that we have used to develop the application. In this case we talk about Android, the operating system on which the application can be used, and Android Studio as the development environment chosen for the implementation of it. Also, we explain about types of Web Services are and what kind we use in this work are exposed. The recognition of text is an important part of the tool and we have focused on explaining how it works and what are the benefits of its use. Finally, we have made a small explanation about the Natural Language Processing and we have mentioned some technologies that currently exist.

- **Chapter 4**: this chapter focuses on the design of the application. We talk about the different steps we follow to achieve a simple, intuitive and finally user friendly design.
- **Chapter 5**: we explain the design of the application architecture. We make a tour of each of the screens explaining what features are presented in each of them. Also, we explain some internal details of the implementation, for example how we connect with the services, how we process the result, how we store certain data, etc.
- **Chapter 6**: after doing the implementation of the application, we made a real user test. In this chapter we explain how this evaluation has been: the objective, the process and the results obtained.
- **Chapter 7**: we wrote the conclusions of the present assignment, as well as some ideas that have emerged during the development and the recommendations of the experts during the evaluation. These ideas and recommendations have been proposed as an option to be developed in the future.
- **Chapter 8**: the contributions made by each one of the project members are exposed.

Capítulo 2

Estado del Arte

"Siempre que te pregunten si puedes hacer un trabajo, contesta que sí y ponte enseguida a aprender cómo se hace"
— Franklin D. Roosevelt

En este capítulo se dará una visión sobre la inclusión y los tipos de accesibilidad digital que existen para personas con discapacidad cognitiva. Además, veremos cómo el uso de las tecnologías permiten hacer una vida más llevadera a quien sufre algún tipo de discapacidad.

2.1. Discapacidad cognitiva: lenguaje y comunicación

El lenguaje y la comunicación son esenciales para el ser humano, para relacionarse y participar en la sociedad. No obstante, las personas que poseen algún tipo de discapacidad cognitiva o intelectual pueden tener serios problemas para interactuar y comunicarse. Esto supone que, a menudo, sufran estados de frustración al no tener recursos suficientes para poder transmitir sus deseos o sentimientos. Es por ello que, a lo largo de los años, se han creado algunas alternativas de comunicación y adaptación del lenguaje para mejorar la inclusión de estas personas.

Según la Asociación Americana de Psiquiatría (2013) cuando hablamos de discapacidad cognitiva debemos tener presente que es una disminución en las habilidades cognitivas e intelectuales del individuo. La discapacidad intelectual no es una enfermedad sino un estado que una persona llevará consigo durante toda su vida. Esta se considera un trastorno intelectual que se manifiesta cuando aparecen síntomas de déficit de atención, de memoria o del lenguaje. Entre estas discapacidades podemos destacar:

- Trastorno del Espectro Autista (TEA): se trata de un trastorno en el desarrollo del individuo que sufre dicha patología, llegando a provocar dificultades en aspectos como el lenguaje y la comunicación. Esto implica que haya dificultades a la hora de relacionarse con otras personas o la imposibilidad de la creación de vínculos afectivos.
- Síndrome de Down: se trata de un trastorno que hace que el individuo sufra tanto irregularidades físicas como retraso mental y social. Entre los afectados por síndrome de Down algunos de los problemas más destacables que suelen presentar son: comportamiento impulsivo, bajo nivel de atención y una capacidad de aprendizaje baja.

A continuación, hablaremos sobre las diversas alternativas que existen actualmente y que hacen posible que personas con distintas capacidades puedan llevar una mejor calidad de vida.

2.2. Sistemas Aumentativos y Alternativos de Comunicación (SAAC)

Cuando hablamos de *comunicación aumentativa y alternativa* (ARASAAC: Gobierno de Aragón, s.f) nos referimos a todas las modalidades de comunicación, aparte del habla, que son utilizadas para expresar ideas, pensamientos, sentimientos, deseos, necesidades, etc. En nuestro día a día, sin darnos cuenta, empleamos este tipo de comunicación cuando hablamos con otras personas y usamos gestos, expresiones faciales, imágenes o incluso la propia escritura. Los SAACs son un conjunto de sistemas utilizados para sustituir o reforzar la forma de comunicación cuando no hay un desarrollo adecuado del habla. Por tanto, si hablamos de personas con dificultades de comprensión del lenguaje o dificultades comunicativas debemos hacer un uso especial de estos sistemas alternativos y aumentativos de comunicación.

Es por ello por lo que el objetivo fundamental de los SAACs es aumentar (aumentativos) la capacidad de comunicación y/o compensar (alternativos) las dificultades del lenguaje favoreciendo así la capacidad de expresión. Gracias a su uso, personas con discapacidades cognitivas pueden mejorar su calidad de vida ya que les permiten desarrollar una autonomía personal y favorecen su autoestima.

Existen algunos *sistemas de símbolos* que se incluyen dentro de los recursos utilizados por los SAACs. Estos se clasifican en:

- *Símbolos gestuales*: dentro de los símbolos gestuales se pueden utilizar desde gestos que usamos diariamente hasta signos manuales, conocidos como lenguaje bimodal o lenguaje signado.

- Por otro lado, existen los *símbolos gráficos*. Estos requieren el uso de recursos tecnológicos (ordenadores o *tablets* con programas especiales o comunicadores de habla artificial); o también el uso de recursos no tecnológicos (tableros y libros de comunicación). Dentro de este tipo de recurso podemos tener desde sistemas sencillos tales como un dibujo, una imagen o una fotografía, hasta otros sistemas más complejos como los sistemas pictográficos o la propia escritura.

Dentro de los símbolos gráficos podemos encontrar los sistemas pictográficos, en los cuales vamos a profundizar por la orientación de este trabajo.

2.2.1. ¿Qué es un pictograma?

Según Bustos (2010) "un pictograma es un dibujo convencionalizado que representa un objeto de manera simplificada y permite transmitir, de este modo, una información también convencionalizada. Los pictogramas son independientes de cualquier lengua particular porque no representan palabras sino realidades". En la Figura 2.1 se muestra un ejemplo de estos.

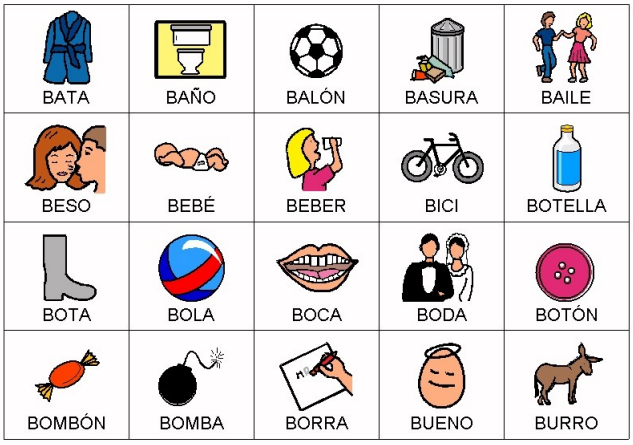


Figura 2.1: Ejemplo de pictogramas. Fuente: Red Cenit

2.2.2. Sistemas pictográficos

En la vida de las personas que poseen diversidad funcional, a veces, es imprescindible el uso de pictogramas. Dichos pictogramas deben tener un diseño simple, coherente y homogéneo. Como se ha expuesto en el apartado anterior, dentro de los SAACs podemos encontrar completos sistemas pictográficos los cuales están compuestos por un gran repertorio de pictogramas.

Actualmente existen diversos tipos de sistemas pictográficos. Veamos algunos ejemplos de los más utilizados.

2.2.2.1. Sistema Bliss

El sistema Bliss (Belloch, s.f) fue desarrollado por Charles Bliss en 1949. Este sistema también es conocido como sistema logográfico ya que se compone fundamentalmente de dibujos geométricos tales como círculo, cuadrados o triángulos, y signos universales tales como números, flechas en distintas direcciones, signos de puntuación, entre otros. Actualmente este sistema está compuesto por 2000 símbolos.

Este SAAC presenta una gran característica que lo diferencia del resto, ya que pueden crearse nuevos símbolos a partir de la combinación de símbolos ya existentes. Gracias a este proceso, expuesto en la Figura 2.2, podemos tener un vocabulario más amplio sin la necesidad de un gran número de símbolos.

Figura 2.2: Proceso de creación de nuevos símbolos usando el sistema Bliss (Belloch, s.f)

2.2.2.2. Símbolos Pictográficos de Comunicación (SPC)

En 1981, Roxana Mayer Johnson elaboró los Símbolos Pictográficos de Comunicación conocidos como SPC (o PSC por sus siglas en inglés *Picture Communication Symbols*). Un SPC es un sistema basado en símbolos pictográficos que por lo general son dibujos sencillos y fáciles de hacer. Es por esto que este SAAC va dirigido especialmente a personas con un bajo nivel de escolarización o que poseen algún tipo de discapacidad intelectual. Esto es así porque los dibujos están diseñados de tal forma que con solo mirarlos sabemos su significado, sin la necesidad de un aprendizaje previo.

Hoy en día existen aproximadamente 3000 iconos que conforman el SPC (Belloch, s.f), aunque también se pueden añadir iconos propios de la cultura de cada región. Estos símbolos se dividen en categorías diferentes y, a su vez, estas categorías se diferencian en colores que ayudan a los usuarios a identificar su significado. Podemos observar esta clasificación en la Figura 2.3.

- Amarillo: para personas o pronombres.
- Verde: para verbos.
- Azul: para adjetivos o adverbios.
- Naranja: para nombres (los que no están incluidos en otras categorías).
- Blanco: para artículos, preposiciones, conjunciones.
- Morado o Rosa: para palabras que se utilizan en las interacciones sociales.

Figura 2.3: Clasificación de pictogramas que conforman el SPC (Belloch, s.f)

2.2.2.3. Minspeak

Este sistema pictográfico (Lloréns Macián, 2006) fue creado por Bruce Baker en 1982 con el objetivo de estimular y acelerar los procesos de comunicación. A diferencia de los dos sistemas pictográficos vistos anteriormente,

Minspeak presenta algunas características diferenciativas relevantes. Sus iconos o pictogramas no tienen un significado preestablecido, sino que pueden tener varios significados dependiendo de la secuencia que se utilice. En la Figura 2.4 podemos observar un ejemplo. En este ejemplo vemos que el icono con forma de casa combinado con otros pictogramas puede tomar distintos significados. Por tanto, podemos decir que el primer pictograma seleccionará el tema y el segundo la idea específica.

Figura 2.4: Ejemplos de pictogramas usando Minspeak (Belloch, s.f)

Minspeak ofrece una gran facilidad para su aprendizaje, ya que se adapta a la capacidad natural de las personas de asociar varios significados a una sola imagen. Además, fomenta un mejor desarrollo del habla ya que proporciona un vocabulario más amplio y la posibilidad de crear frases completas.

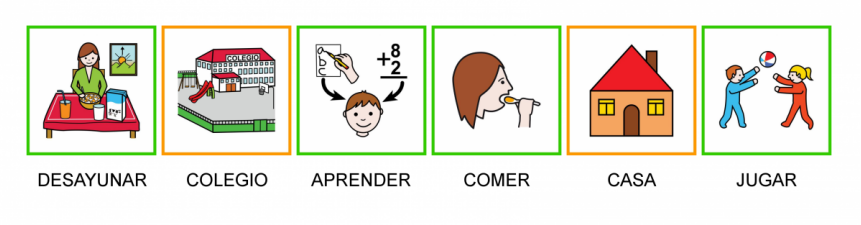
2.2.2.4. ARASAAC

ARASAAC[1] es otro sistema pictográfico. Este sistema, junto con el SPC, son los sistemas que más se usan en gran parte del territorio español. Está desarrollado por el Portal Aragonés de la Comunicación Alternativa y Aumentativa. Surge con la idea de que todas las personas con distintas capacidades que necesiten hacer uso los SAACs, para mejorar la forma de comunicación, tengan acceso libre a su amplio repertorio de pictogramas.

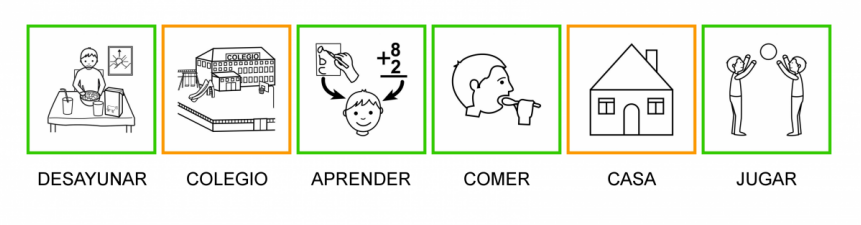
El Portal Aragonés ofrece, bajo la licencia Creative Commons, una amplia bases de datos en la cual no solo dispone de pictogramas, sino que ofrece un gran catálogo con una gran variedad de recursos. En él podemos acceder a pictogramas en color y en blanco y negro disponibles en varios idiomas, fotografías, vídeos, así como el catálogo de la Lengua de Signos Española (LSE). En las Figuras 2.5a y 2.5b podemos ver algunos ejemplos de pictogramas ARASAAC a color y en blanco y negro, respectivamente. Además, en la Figura 2.5c se muestra un ejemplo de traducción a lenguaje de signos.

En el estudio "Características formales y transparencia de los símbolos pictográficos de ARASAAC", realizado por Cabello y Bertola (2015), se han analizado las características de los pictogramas ARASAAC. Algunas de las

[1] Para más información acceder a `http://www.arasaac.org/aac.php`



(a) Pictogramas ARASAAC en color



(b) Pictogramas ARASAAC en blanco y negro

(c) Traducción a lenguaje de signos

Figura 2.5: Ejemplos de pictogramas ARASAAC y traducción a lenguaje de signos

características estudiadas son realismo, ambigüedad, eficiencia, transparencia e iconicidad[2], y se han comparado con otros sistemas pictográficos tales como el SPC o Bliss, comentados en apartados anteriores. Gracias a este estudio se ha llegado a la conclusión de que los pictogramas ARASAAC tienen un adecuado grado de transparencia. Esto quiere decir que el significado de cada pictograma es muy claro y conciso. Además, presentan una alta iconicidad, ya que el signo no verbal y significado del pictograma en sí tienen una relación evidente, lo que significa que cumplen con su función comunicativa.

Después de haber hecho un estudio de los distintos pictogramas que existen actualmente, hemos decidido usar pictogramas ARASAAC en nuestro proyecto.

[2] Según el artículo, con iconicidad se refiere al grado de relación que un individuo hace entre un símbolo y su referente.

2.2.3. Aplicaciones basadas en pictogramas

Afortunadamente, hoy en día existen una gran variedad de proyectos y aplicaciones que promueven el uso de los Sistemas Aumentativos y Alternativos de Comunicación, y más concretamente de los pictogramas. A continuación, expondremos ejemplos de algunas de estas aplicaciones.

2.2.3.1. AraWord

AraWord (Baldassarri et al., 2013) es un software informático de libre distribución. Consiste en un traductor de texto que permite la escritura simultánea de textos y pictogramas. Este proceso facilita la adaptación de texto a pictogramas para personas con dificultades en el desarrollo del lenguaje. Esta herramienta ofrece la posibilidad de crear materiales como itinerarios, agendas, cuentos, entre otras. De AraWord hay que destacar que es muy fácil su uso y además utiliza los pictogramas ARASAAC por defecto, aunque se pueden añadir pictogramas personalizados. En la Figura 2.6 se expone un ejemplo de traducción de texto.

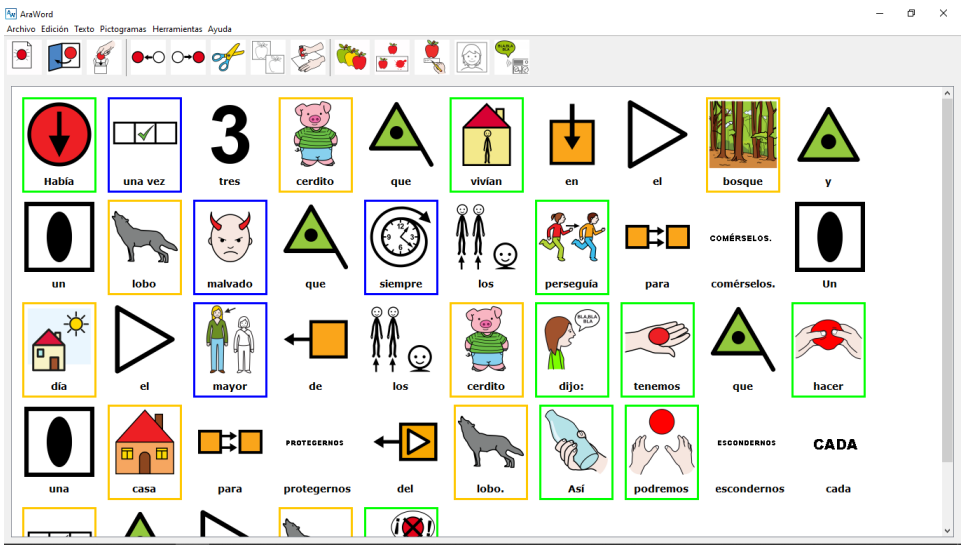


Figura 2.6: Ejemplo de traducción del inicio del cuento infantil "Los tres cerditos" a pictogramas usando el sotfware AraWord.

2.2.3.2. PICTAR

PICTAR es una herramienta informática que tiene como objetivo principal facilitar los trabajos que requieren el uso de pictogramas en el ámbito educativo. Esta herramienta fue desarrollada por Martín Guerrero (2018)

(a)

(b)

Figura 2.7: Ejemplo de uso de PICTAR.

como Trabajo de Fin de Máster. PICTAR se divide en dos partes. Por un lado se presenta como un servicio web, del que hablaremos en apartados posteriores, para hacer la traducción *texto-pictogramas* y, por otro lado, está la página web (`http://hypatia.fdi.ucm.es/pictar/`) donde se pueden crear y modificar los materiales utilizando pictogramas ARASAAC. Si accedemos a la página web podremos ver que está dividida en tres partes: *Traductor* (Figura 2.7a), *Buscador* y *Editor* (Figura 2.7b).

- Editor: esta es la parte principal de la aplicación. En esta sección se permite generar y modificar los materiales a gusto del usuario. Existen una serie de opciones, que podemos ver en la Figura 2.7b en las que el propio usuario podrá modificar el número de cuadrículas o elementos y

el número de columnas para generar el material. En estas cuadrículas es donde se colocarán los pictogramas para crear los materiales escolares. Además, se puede cambiar el color de los pictogramas a blanco y negro, así como añadir una descripción o nombre a cada uno. También, existe la opción de exportar o importar los tableros. Es decir, si estamos trabajando y editando el tablero de pictogramas y no terminamos el trabajo, podremos exportarlo a nuestro equipo para, posteriormente, importarlo a la aplicación y seguir editando.

- Buscador: en esta parte aparece una barra de búsqueda en la cual el usuario podrá poner cualquier palabra en español para que el sistema busque sus correspondientes pictogramas, ya que una palabra puede tener asociado más de un pictograma. En la Figura 2.7b se muestra un ejemplo de todos los pictogramas asociados a la palabra *jugar*. Un vez encontrado el pictograma buscado se podrá arrastrar y poner en cualquiera de los espacios disponibles en la parte *Editor*.

- Traductor: en este área de la página web aparece, también, una barra de búsqueda en la que se podrá escribir cualquier frase en castellano que será traducida a pictogramas. Una vez hecha la traducción, si alguna palabra tiene más de un pictograma, el usuario podrá cambiar de pictogramas usando las flechas que aparecen en cada uno de los mismos. De la misma forma que el *Buscador*, se podrá seleccionar un pictograma y arrastrar hasta la zona del *Editor*. También si pulsamos el botón que aparece debajo de la traducción a pictogramas, como se muestra en la Figura 2.7a, se copiará automáticamente la frase traducida a la zona del *Editor*.

2.2.3.3. DictaPicto

DictaPicto (Juan et al., 2017) es una aplicación gratuita para móviles que permite hacer una traducción en tiempo real de un texto escrito o un mensaje de voz a pictogramas, imágenes o fotos personalizadas por el usuario. Fue desarrollada por BJ-Adaptaciones y Doble Equipo Valencia en 2016, y publicada por Fundación Orange. El objetivo principal de esta aplicación es mejorar la capacidad de comunicación y facilitar la participación e interacción con el entorno de las personas con Trastornos del Espectro Autista (TEA) o personas que usan este tipo de SAAC.

La principal funcionalidad de DictaPicto es secuenciar y anticipar las actividades de la vida cotidiana. Nos permite grabar un mensaje de voz o escribir un texto que será traducido automáticamente a pictogramas. DictaPicto ofrece el uso de pictogramas ARASAAC, aunque también se pueden añadir imágenes personalizadas. Una vez realizada la traducción se podrán modificar los pictogramas e imágenes sin tener que modificar el mensaje.

DictaPicto, además, ofrece una opción para que el usuario pueda diferenciar entre una orden o norma, una pregunta o una afirmación, facilitando así la comprensión del lenguaje.

- Norma: como se muestra en la Figura 2.8a, si al comienzo de la frase que se pretenda traducir se dice o se escribe la palabra *norma*, se ayudará al usuario a comprender que lo que viene después del pictograma norma es una orden.

- Anticipación: con la misma idea que *Norma*, se escribirá o se dirá la palabra *anticipación*. De esta forma ayudará al usuario a comprender que, después del pictograma *anticipación*, vendrán una serie de actividades que se van a ejecutar en el futuro. Podemos ver un ejemplo en la Figura 2.8b.

- Pregunta: este pictograma se utilizará para que el usuario comprenda que la frase (pregunta) requiere de una respuesta, y para entender mejor el uso del signo de interrogación al final de la pregunta. Un ejemplo de ello podemos verlo en la Figura 2.8c.

(a) Normas en DictaPicto

(b) Anticipación en DictaPicto

(c) Pregunta en DictaPicto

Figura 2.8: Ejemplos de uso de DictaPicto.

2.3. Lectura Fácil

El acceso a la información es un derecho universal de todos los seres humanos. Sin embargo, no todos somos iguales ni tenemos la misma capacidad o desarrollo cognitivo y, por eso, necesitamos diferentes formas o soluciones que nos faciliten el acceso a la información. La *Lectura Fácil* es una de ellas. A pesar de que, leyendo estas dos palabras, "lectura fácil", viene implícito el significado, queremos dar una definición más amplia. La Lectura Fácil es una forma de adaptar la información escrita para que esta sea más sencilla de leer y entender por personas con dificultades de comprensión lectora; hablamos de personas con discapacidad intelectual, personas con un nivel bajo de escolarización, personas mayores e incluso personas que están aprendiendo un idioma.

Es posible hacer esta adaptación para cualquier información. No obstante, se deben seguir unas directrices para que un texto quede adaptado a Lectura Fácil.

2.3.1. Directrices para materiales de la Lectura Fácil

En 1997 se establece la definición de Lectura Fácil con la publicación de "Directrices para materiales de lectura fácil" de la Federación Internacional de Asociaciones de Bibliotecarios y Bibliotecas (IFLA[3]) (Tronbacke et al., 2010). Este documento supuso la primera guía para elaborar textos en Lectura Fácil aplicando una serie de pautas. Algunas de las pautas esenciales a seguir para adaptar un texto son:

- Hacer uso de frases cortas y utilizar un lenguaje sencillo, simple y directo.
- Evitar frases o preguntas en negativo.
- Intentar usar la voz activa en vez de la pasiva.
- No utilizar metáforas o comparaciones u otras figuras literarias que puedan resultar confusas.
- Se recomienda usar frecuentemente preposiciones y conjunciones.
- Escribir un único mensaje por cada frase. Ser claro, conciso y directo.
- Evitar el uso de aquellas palabras que se consideran difíciles de entender tales como los tecnicismos o abreviaturas.

[3] Para más información consultar el sitio web oficial de Federación Internacional de Asociaciones de Bibliotecarios y Bibliotecas (IFLA) `https://www.ifla.org/`

- Las palabras que se utilicen deben ser precisas. Debe evitarse el uso de palabras genéricas como “cosa”, “objeto”, etc.
- Cualquier frase o idea que se considere innecesaria deberá evitarse.
- No abusar del uso de la mayúscula. Las mayúsculas producen una mayor dificultad para leerse.
- Con respecto al uso de signos de puntuación:
  - El punto será el signo ortográfico fundamental en la Lectura Fácil. Por lo general se usará el punto y aparte frente al punto y seguido. De esta forma se separan y diferencian mejor las ideas en el texto.
  - La coma se usará para separar elementos de una lista o para enumerar. No obstante, es mejor limitar su uso frente al punto.
  - Los dos puntos se aplican para introducir algún diálogo o para completar ideas de una frase.
  - Es recomendable evitar el uso de signos poco habituales tales como paréntesis, corchetes, entre otros.
- Evitar el uso de números o denominaciones matemáticas. Si no queda más remedio mejor escribir el número con letras. Por ejemplo: en vez de 5, escribiremos *cinco*.
- Las imágenes, pictogramas, gráficos o los símbolos son de gran apoyo al texto siempre y cuando sean fáciles de entender y estén estrechamente relacionadas con el mismo.

Proponemos como pequeño ejemplo de Lectura Fácil el principio de la obra en prosa más importante: “Don Quijote de la Mancha” de Miguel de Cervantes Saavedra. Seguramente habremos leído alguna vez o nos sonará este pequeño fragmento:

> *“En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los viernes, algún palomino de añadidura los domingos, consumían las tres partes de su hacienda. El resto della concluían sayo de velarte, calzas de velludo para las fiestas con sus pantuflos de lo mismo, los días de entre semana se honraba con su vellori de lo más fino. Tenía en su casa una ama que pasaba de los cuarenta, y una sobrina que no llegaba a los veinte, y un mozo de campo y plaza, que así ensillaba el rocín como tomaba la podadera.”*

Como bien hemos explicado anteriormente, a la hora de hacer la adaptación podríamos incluir imágenes que estén relacionadas con el texto o bien hacer definiciones de palabras que se consideren difíciles de entender. Una traducción o adaptación a texto fácil podría ser la siguiente (Anula y Belinchón, 2010):

> En un pueblo de la Mancha,
> de cuyo nombre no quiero acordarme,
> vivió no hace mucho tiempo un **hidalgo**.
>
> Nuestro hidalgo se llamaba Alonso Quijano.
> Tenía muchos años y era muy delgado.
> Don Alonso poseía un caballo flaco,
> unas tierras y una casa muy grande.
> El hidalgo vivía con su joven sobrina
> y una criada.

Si nos fijamos en el texto anterior la palabra *hidalgo* aparece en negrita. En el documento de Anula y Belinchón (2010) donde está escrita esta adaptación aparece el significado de *hidalgo* como: "Un hidalgo era una persona que había heredado tierras y vivía sin tener que trabajar. Era un noble". Además, aparece una imagen que ayuda a tener una mejor comprensión del texto y que podemos ver en la Figura 2.9.

Figura 2.9: Imagen que aparece en el documento adaptado a Lectura Fácil del Quijote de la Mancha (Anula y Belinchón, 2010)

2.3.2. Proyectos relacionados con la Lectura Fácil

Hemos podido observar que son muchos los beneficios que aporta el uso de la Lectura Fácil. Consideramos que su uso es necesario ya que fomenta la inclusión social haciendo posible el acceso a la información a cualquier persona independientemente de su nivel de cognición. Además, promueve el hábito de la lectura, elimina las barreras en la comunicación y mejora la calidad de vida de las personas.

En la actualidad, existen diversos proyectos que promueven el uso de la Lectura Fácil. Han sido creados, en su mayoría, para facilitar el acceso de la información a toda persona que lo necesite. A continuación, expondremos algunos ejemplos.

2.3.2.1. Simplext

La simplificación de texto es una tecnología de procesamiento de lenguaje natural, la cual explicaremos en apartados posteriores, que puede usarse para generar textos que se adapten a las necesidades de las personas. La mayoría de los proyectos dedicado al área de la simplificación de textos han centrado su investigación en el idioma inglés. Es por esta razón que surge Simplext, un proyecto dedicado a la simplificación automática de texto en español.

Simplext (Saggion et al., 2011) es un proyecto del Ministerio de Industria, Energía y Turismo, mediante el Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica. Es una herramienta *software* que consiste en la simplificación automática de textos reduciendo la complejidad semántica y léxica, facilitando así el acceso a la lectura a personas con discapacidades intelectuales o que tienen un bajo nivel de escolarización. Para la simplificación de los textos, Simplext se basa en los principios de Lectura Fácil comentados en el apartado anterior. Para acceder a esta herramienta podemos visitar la página web de la demo de Simplext (`http://simplext.taln.upf.edu/`). Su funcionalidad es muy sencilla, basta con copiar el texto que queramos adaptar y la simplificación del mismo se hará automáticamente como se muestra en la Figura 2.10.

2.3.2.2. PorSimples

A menudo, la forma que tenemos de escribir un texto puede suponer una barrera para muchas personas. *PorSimples* (Aluisio et al., 2010) es otro proyecto relacionado con la simplificación y adaptación de texto, en este caso para textos escritos en el idioma portugués.

El objetivo principal es desarrollar tecnologías que usen procesamiento de lenguaje natural y estén relacionadas con la adaptación de texto. PorSimples

Figura 2.10: Ejemplo de simplificación de texto usando la herramienta Simplext

proporciona dos herramientas diferentes que comentaremos a continuación:

- Por un lado está la *simplificación automática de texto*. Esta parte tiene como objetivo la reducción de la complejidad léxica y/o sintáctica de un texto, preservando el contenido original del mismo. De esta forma facilita a lectores con bajos niveles de escolarización la comprensión del contenido de un texto. Para esto se ha creado la herramienta FACILITA que consiste en un *plugin* para navegadores web que permite resumir y simplificar todo el contenido que aparece en la web.
- Por otro lado, se ha creado la herramienta SIMPLIFICA para la *elaboración de texto*. Dicha herramienta consiste en un editor de texto en el que se puede generar contenido adaptado. Este editor, además, proporciona definiciones o sinónimos de las palabras utilizadas para facilitar y mejorar la creación de contenido.

2.3.2.3. ReadIt

ReadIt[4] es una aplicación desarrollada por Jiménez Corta (2018) como Trabajo de Fin de Grado. Consiste en un *plugin* o extensión para Google Chrome donde se ofrecen distintas opciones que se adaptan a las necesidades de los usuarios. Se creó con el objetivo de que cualquier persona, independientemente de su nivel de cognición, pudiera acceder y navegar por cualquier página web con total independencia.

Una vez descargada y activada la extensión en Google Chrome, veremos que aparece una barra en la parte superior del navegador, que estará visible

[4] Podemos acceder la aplicación en el siguiente enlace: `https://chrome.google.com/webstore/detail/readit/hjjfoaojoajlgogaadpmenblklambekl`

en todas las páginas por las que se navegue. En esta barra aparecerán todas las opciones que ofrece la aplicación. Por tanto, si el usuario que navega no entiende una palabra o una frase dentro de un texto de la página web, podrá pulsar en las distintas opciones para tener una mejor comprensión del contenido. En la Figura 2.11 podemos ver las opciones que ofrece ReadIt:

Figura 2.11: Barra de opciones de ReadIt

- *Búsqueda*: el usuario, además de seleccionar las palabras o frases en la página web, podrá usar la opción de búsqueda para buscar alguna palabra que no comprenda.
- *Definiciones*: cuando seleccionamos una palabra, esta opción nos mostrará todos los significados de la misma. De la misma forma funcionan las opciones *Palabras parecidas* y *Palabras contraria*, mostrando sinónimos y antónimos de la palabra, respectivamente.
- *Pictogramas*: mostrará las posibles traducciones a pictogramas de una frase o una palabra.
- Para las opciones *YouTube* y *Wikipedia* se abrirá una ventana en la que hará una búsqueda en estas dos páginas.
- *Resumen*: en caso de que el usuario no comprenda un texto completo, podrá seleccionar esta opción y se mostrará un resumen del mismo.
- *Leer en voz alta*: al pulsar esta opción, la aplicación leerá en voz alta la palabra o el texto seleccionado.

Capítulo 3

Tecnologías Utilizadas

"No inventé nada nuevo. Simplemente junté
los descubrimientos de otros hombres
que trabajaron durante siglos"
— Henry Ford

En este capítulo hablaremos sobre las distintas tecnologías utilizadas para el desarrollo del presente trabajo. Expondremos los motivos por los cuales hemos decidimos usar unas tecnologías frente a otras y hablaremos sobre los orígenes de las mismas.

3.1. Android

Android[1] es un sistema operativo de código abierto, diseñado para dispositivos móviles con pantalla táctil, cuyo núcleo está basado en el kernel de Linux. Android tiene la capacidad de soportar diversos tamaños de pantallas y resoluciones. La interfaz de usuarios de Android pretende ser sencilla e intuitiva, y está pensada para su manipulación directa, es decir, al poseer una pantalla táctil el usuario podrá pulsar la pantalla, seleccionar elementos, arrastrarlos y soltarlos. Android permite modificar a gusto del usuario la pantalla principal, pudiendo cambiar fondos, añadir widgets y accesos directos de las aplicaciones instaladas en el dispositivo. Posee un sistema de notificaciones que avisa al usuario cuando recibe correos, mensajes, etc. El código fuente fue liberado por Google bajo la licencia Apache 2.0, lo que significa que cualquier persona puede modificar el código con total libertad aportando así sus conocimientos.

[1] Para más información acceder a `https://source.android.com/`

3.1.1. Orígenes de Android

Se podría definir un *smartphone* o teléfono inteligente como un dispositivo en el que se integran las funcionalidades de un teléfono móvil junto con las funcionalidades de los antiguos asistentes personales o PDAs, es decir, podemos realizar llamadas, enviar o recibir mensajes, revisar el correo, navegar por internet, hacer uso de la alarma o hacer fotos, entre otras, usando un único dispositivo. Se dice que el primer *smartphone* apareció en 1992 (NeoTeo, 2012) con el lanzamiento del IBM Simon, pero no fue hasta 2007, con la llegada del iPhone, cuando Apple revolucionó el concepto de *smartphone* presentando un dispositivo con pantalla táctil sin teclado físico, acelerómetros para detectar la posición del mismo y capacidades multimedia, entre otras. Estos dispositivos incluían un sistema operativo llamado iPhone OS y que, en versiones posteriores, pasó a llamarse iOS. El iOS de Apple no es el primer sistema operativo de los *smartphones* pero sí marcó el comienzo de una nueva era en el mundo de los teléfonos inteligentes, y a partir de este momento es cuando el Android de Google toma partido convirtiéndose así en el sistema operativo para *smartphones* más popular en el mundo. Android ha evolucionado significativamente desde sus inicios cuando se lanzó por primera vez en un dispositivo T-Mobile de HTC en 2008. Sin embargo, la historia de Android se remonta a mucho antes y era muy distinto a como lo conocemos ahora.

La compañía Android Inc. (Callaham, 2018) fue creada por Andy Rubin, Rich Miner, Nick Sears y Chris White en 2003. El principal objetivo de la compañía era desarrollar un sistema operativo para cámaras digitales. Pronto se dieron cuenta de que el mercado de los sistemas operativos de cámaras digitales quizás no era tan grande y, por tanto, desviaron su atención hacia los teléfonos inteligentes. Meses después se lanzó Android como un sistema operativo para teléfonos haciendo frente a los líderes del mercado en telefonía en ese momento Symbian y Microsoft Windows Mobile. En 2005 (Ruiz Martín et al., 2014) Android Inc. pasa a ser propiedad de Google y esto supone un gran capítulo en la historia de Android. Varios de sus miembros fundadores se quedaron para continuar desarrollando el sistema operativo con sus nuevos propietarios. Se tomó la decisión de utilizar Linux como base del sistema operativo. Google tenía la idea de crear un sistema operativo y estaba trabajando en un prototipo muy parecido al de un teléfono móvil BlackBerry. En 2007, como se mencionó anteriormente, Apple lanzó el primer iPhone y esto supuso un antes y un después en el mundo de la telefonía móvil. El iPhone apareció como un dispositivo con pantalla táctil. Esto supuso un contratiempo para Google que inmediatamente tuvo que replantear el diseño del dispositivo, así como la idea del sistema operativo, para poder competir contra el iPhone. Ese mismo año se creó la Alianza para los Dispositivos Móviles Abiertos, conocida como OHA por sus siglas en inglés *Open Handset*

Alliance. Esta organización, liderada por Google, reunía en sus comienzos a una treintena de empresas. Entre ellas se incluían empresas de fabricantes de teléfonos como HTC, Motorola y Samsung y fabricantes de chips como Qualcomm, Nvidia y Texas Instruments; y operadores como T-Mobile, entre otros. Junto con el anuncio de la formación de la OHA se presentó Android. En ese instante, Android era la primera plataforma completa para dispositivos móviles basada en el kernel de Linux. Esta plataforma incluía un sistema operativo, aplicaciones y una interfaz de usuario. El objetivo fundamental de la OHA era, por tanto, favorecer la innovación en los dispositivos móviles proporcionando una plataforma abierta y completa. En 2008 salió el HTC Dream, el primer *smartphone* comercializado con Android.

Android ha recorrido un largo camino y ha ido evolucionando considerablemente desde sus humildes comienzos, cuando solo se propuso como una idea para crear sistemas operativos para cámaras digitales, hasta convertirse en el sistema operativo para móviles líder en todo el mundo. En la gráfica de la Figura 3.1 se puede observar el crecimiento que ha tenido Android desde sus comienzos hasta la actualidad. Han ido apareciendo nuevas versiones del sistema operativo, desde su primera versión en 2008 (Android 1.0) hasta su versión más actualizada en 2018 (Android 9.0). Pero Android no solo está limitado a dispositivos móviles (*smartphones* o *tablets*) sino que se ha ido extendiendo a otros dispositivos incluyendo Android TV, Android Auto o WearOS, este último para relojes inteligentes o *smartwatches*.

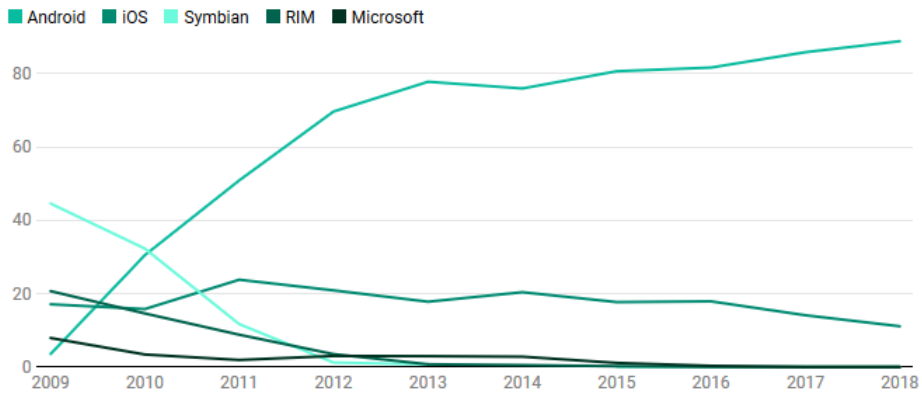


Figura 3.1: Evolución de la cuota de mercado de los sistemas operativos para móviles. Fuente: Xataka Móvil

3.1.2. Arquitectura de Android como Sistema Operativo

La arquitectura de Android (Universitat Politècnica de València, s.f) está compuesta por cinco capas diferentes. En la Figura 3.2 podemos observar

los principales componentes de la plataforma Android. Cada una de estas capas ofrece servicios a las capas que están en niveles superiores, así como también estas propias capas pueden obtener los servicios de las capas de niveles inferiores. Viendo la Figura 3.2 explicaremos las capas de abajo a arriba.

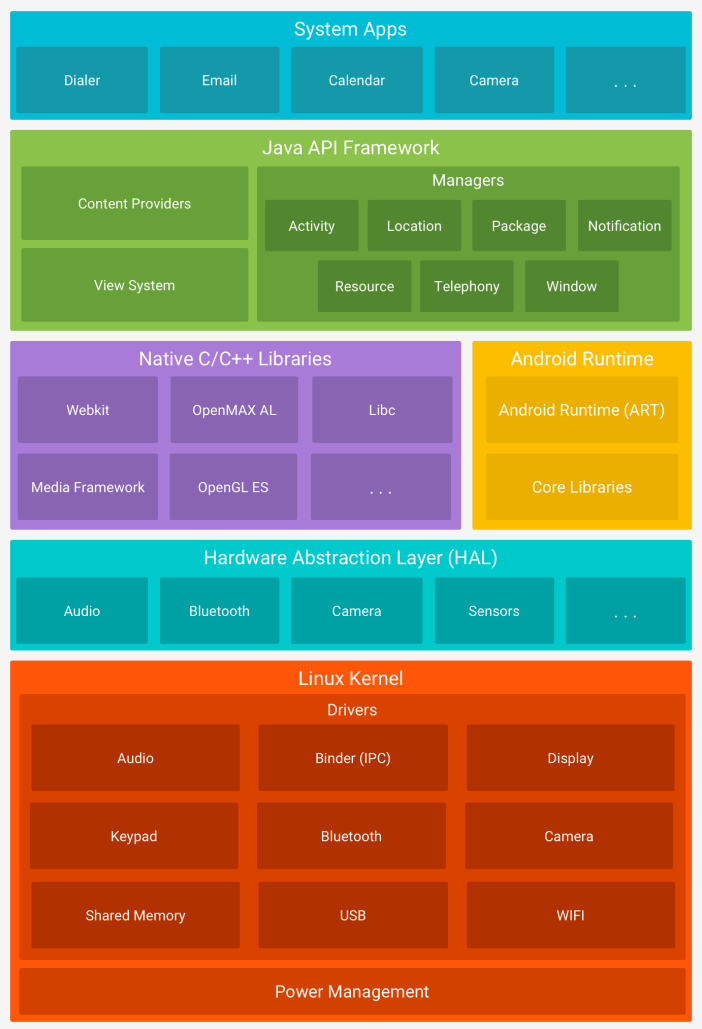


Figura 3.2: Arquitectura de Android. Fuente: Android Developers

- Núcleo de Linux:

  El núcleo de Android está basado en el *kernel* de Linux. En esta capa se aprovechan los servicios de seguridad, multiproceso, manejo de memoria, etc. proporcionados por el *kernel*. El *kernel* actúa como intermediario entre el *hardware* y el resto de las capas y es aquí donde

se encuentran todos los *drivers* necesarios para manejar el dispositivo. Sin embargo, el desarrollador no puede acceder directamente a estos *drivers* sino que debe hacerlo a través librerías (pertenecientes a las capas superiores) que se montan sobre el *kernel*.

- Capa de Abstracción de *Hardware* o HAL:

  La capa de abstracción de *hardware*, o HAL por sus siglas en inglés *Hardware Abstraction Layer*, se encarga se acceder al *hardware* a través de los *drivers*, es decir, evita que las aplicaciones del sistema accedan directamente al *hardware*. Esta capa permite, además, que las aplicaciones sean independientes del *hardware* ya que abstraen la información de componentes como cachés, buses de E/S, etc.

- Librerías nativas:

  Como se observa en la Figura 3.2 esta capa está al mismo nivel que el entorno de ejecución de Android. Aquí se incluyen diversas librerías programadas en C o C++ que son las que proporcionan las principales funcionalidades del sistema operativo Android y de esta manera facilitan la reutilización del código. Dentro de las librerías incluidas destacamos algunas como: *Surface Manager*, *WebKit*, *SQLite*, *SSL* y *Secure Socket Layer*, entre otras.

- *Runtime* de Android:

  El entorno de ejecución de Android se sitúa en el mismo nivel de las librerías nativas de Android y es aquí donde se encuentra la DMV (*Dalvik Virtual Machine*). Esta máquina virtual está basada en el concepto de máquina virtual de Java y fue implementada por Google para resolver los problemas de limitaciones de memoria y procesador que albergaban los dispositivos móviles, ya que no era posible utilizar una máquina virtual Java estándar.

  La máquina virtual *Dalvik* no es una JVM (*Java Virtual Machine*), sino que es una máquina creada desde cero basada en la JVM. Esto se hizo así con el objetivo de evitar los problemas de licencia. No obstante, la DMV permite la ejecución de aplicaciones programadas en Java. Como todos los dispositivos Android tienen integrada esta máquina *Dalvik* se podrán crear aplicaciones para Android sin tener en cuenta el tipo de dispositivo en el que se ejecutaren dichas aplicaciones.

- Entorno de aplicación:

  Esta capa, también conocida como *framework* de aplicaciones, supone toda la estructura sobre la que se montan todas las aplicaciones

Android, es decir, contiene todos los componentes y librerías necesarias para crear una aplicación Android. Dentro de las funcionalidades más importantes que ofrece podemos encontrar el uso de las alertas de sonido, la cámara, el acceso al sistema de notificaciones, acceso a la localización del dispositivo o al almacenamiento del mismo, entre otras.

- Aplicaciones:

  En esta última capa encontramos las aplicaciones que vienen por defecto en el dispositivo, así como también aplicaciones creadas por terceros o aplicaciones creadas por nosotros mismos. Para garantizar la seguridad del sistema todas las aplicaciones deben ser ejecutadas en la máquina virtual *Dalvik*.

3.1.3. Android Studio

Android Studio (Nilanchala, 2013) es un entorno de desarrollo integrado, o IDE por sus siglas en inglés *Integrated Development Environment*, que incluye todo lo necesario para crear aplicaciones Android. La primera versión estable de Android Studio salió en 2014. Fue desarrollado por Google y está basado en el *software* de Intellij IDEA. Este entorno de desarrollo integrado se creó como una alternativa a Eclipse. Se buscaba un *software* con una mayor potencia, y que fuese mucho más fácil e intuitivo a la hora de desarrollar aplicaciones. Esta herramienta permite desarrollar aplicaciones, realizar cambios y obtener una vista previa en tiempo real, aumentando la productividad y haciendo que sea más fácil crear aplicaciones atractivas con menos esfuerzo.

Son muchas las características que podemos encontrar si usamos esta herramienta para el desarrollo de aplicaciones. Desde sus inicios hasta el momento se han ido creando nuevas funcionalidades con cada actualización. Una de las características fundamentales es poder visualizar la aplicación a medida que se vaya desarrollando, además del *Android Emulator*, que permite hacer una simulación de distintos dispositivos en los que poder ejecutar la aplicación en desarrollo y probarla sin necesidad de usar el *hardware* directamente.

3.2. Servicios Web

Desde su creación hasta la actualidad se ha ido modificando la definición de servicio web. Según el WC3 (Booth et al., 2004) “un servicio web es un sistema de *software* diseñado para admitir la interacción interoperable

de máquina a máquina a través de una red", es decir, podemos definir un servicio web como un componente al que podemos acceder a través de la red usando protocolos.

En la actualidad existen infinidad de aplicaciones informáticas desarrolladas en diversos lenguajes de programación tales como Java, C++, PHP y en distintas plataformas como Windows, GNU/Linux, entre otras. No es imposible imaginar que la mayoría de estas aplicaciones necesiten, o hayan necesitado en algún momento, establecer una comunicación entre ellas, ya sea por ofrecer algún servicio o por recibirlo. Sin embargo, al estar implementadas en distintos lenguajes y plataformas se hace más difícil establecer esta comunicación. Es aquí es donde entran en juego los servicios web. Estos surgen por la necesidad de crear un modelo estándar de comunicación para que aplicaciones o sistemas tengan la capacidad de comunicarse independientemente de la plataforma o lenguajes de programación en las que estén desarrolladas. El objetivo de esta comunicación entre aplicaciones es poder desacoplar determinadas funcionalidades, pudiendo crear APIs que ofrezcan determinados servicios, para así garantizar la reutilización de código o funcionalidades.

La mayoría de los servicios web se basan principalmente en una arquitectura orientada a servicio conocida como SOA por sus siglas en inglés *Service-Oriented Architecture*. SOA es un modelo de diseño de *software* compuesto por diferentes servicios que interactúan entre sí y además se pueden combinar dando lugar a nuevos servicios con más funcionalidades. La flexibilidad y la versatilidad son las características fundamentales de SOA ya que permite la reutilización y facilita la interacción entre cliente y servidor independientemente de la plataforma o aplicación que utilicen. A continuación, hablaremos de dos de los tipos de servicios web más utilizados actualmente: SOAP y REST.

3.2.1. Servicios Web SOAP

Los servicios web SOAP (Universidad de Alicante, 2012) son un tipo de servicio web. La idea principal de SOAP fue asegurar que las aplicaciones implementadas en diferentes plataformas y lenguajes de programación pudieran intercambiar datos de una manera más sencilla. Para lograr un correcto funcionamiento de un servicio web de este tipo tienen que estar presente una serie de componentes. Estos deben ser completamente independientes del lenguaje de programación utilizado para la implementación del servicio web.

- SOAP. Por sus siglas en inglés *Simple Object Access Protocol* es un protocolo de mensajería escrito en XML para el intercambio de información entre aplicaciones. En otras palabras, podemos decir que es

una manera de definir la estructura que debe seguir la información que se envía y cómo se envía mediante XML. Un mensaje SOAP se codifica como un documento XML. Cada mensaje debe incluir un elemento `<Envelope>` en el que se incluirán otros dos elementos `<Header>` y `<Body>`. El primero es el encabezado que contiene los datos de enrutamiento, es decir, la información del cliente al que se le envía el documento, y el segundo elemento contiene el mensaje. Dentro del `<Body>` se incluye un subelemento `<Fault>` que se utiliza para notificar errores.

- WSDL. No se puede acceder a un servicio web si se desconoce dónde está ubicado y qué hace realmente el servicio. Es por ello por lo que el cliente debe saber dónde reside el servicio web y, además, conocer cuáles son sus funcionalidades para poder hacer uso del mismo. Esto se hace con la ayuda del lenguaje de descripción de servicios web o WSDL por sus siglas en inglés *Web Services Description Language*. El WSDL es un archivo basado en XML que permite especificar a los clientes cómo deberían ser los parámetros de entrada y salida para que puedan acceder a los servicios.

- UDDI por sus siglas en inglés *Universal Description, Discovery and Integration* es un estándar XML utilizado para describir, publicar y encontrar servicios web proporcionado por los proveedores de servicios. Anteriormente comentamos qué eran los WSDL, pero ¿cómo pueden acceder los clientes a esta información? Esto se hace a través de un repositorio (UDDI) que contiene la toda la información referente a estos servicios web.

3.2.2. Servicios Web REST

REST (BBVAOpen4U, 2016) es el acrónimo de *Representational State Transfer* que significa Transferencia del Estado Representacional. Este tipo de servicio web se crea como alternativa a SOAP y, a diferencia de éste, no es un protocolo sino un estilo arquitectónico para crear servicios web. Todos los servicios web basados en la arquitectura REST pueden establecer una comunicación entre dos sistemas para enviar o recibir datos, o realizar operaciones sobre los mismos en cualquier formato (JSON, XML). El sistema REST está basado en el protocolo HTTP para la transmisión de datos. Estos datos se consideran recursos y se accede a ellos utilizando identificadores uniformes de recursos (URI). Para su funcionamiento, REST se basa en los siguientes principios:

- Cliente-servidor: las aplicaciones REST están basadas en la arquitectura cliente-servidor, es decir, hay un servidor y un cliente que realiza peticiones a dicho servidor. El cliente (el usuario) se comunica con el

servidor para acceder a los servicios ofrecidos por el mismo. Esto permite separar las responsabilidades que tiene el cliente y el servidor y, además, aumenta la escalabilidad trabajando de forma independiente.

- Sin estado: el servidor no mantiene ningún estado de los clientes. En cada petición HTTP está contenida toda la información necesaria para ejecutar esta petición y de esta forma se evita que el servidor tenga que recordar un estado previo para ejecutar dicha petición.

- Interfaz uniforme: garantiza la uniformidad en todas las operaciones con las que se trabaja. Los servicios REST proporcionan datos como recursos, con un espacio de nombres consistente. Cuando se hace un envío de datos, el sistema REST aplica una serie de acciones concretas sobre los recursos, tales como GET, para crear nuevos recursos; POST, para leer y/o consultar; PUT, para editar; y DELETE, para eliminar.

- Sistema de capas: REST está compuesto por una arquitectura jerárquica. Los componentes del sistema no pueden conocer la funcionalidad de otros componentes que no pertenecen a su misma capa. Esto garantiza una mejor seguridad y rendimiento en el sistema.

3.2.3. Ejemplos de servicios web REST

Después de haber hecho un estudio sobre los dos principales servicios web que existen actualmente, hemos decidido usar servicios web REST. En este apartado veremos algunos ejemplos de servicios web REST que usaremos para la implementación del presente trabajo.

3.2.3.1. Servicio de traducción texto-picto de PICTAR

En el Capítulo 2 se expusieron ejemplos sobre aplicaciones que estaban basadas en SAACs. Una de estas era PICTAR, una herramienta para elaborar contenidos para personas con TEA. Esta herramienta ha sido creada por Martín Guerrero (2018) como Trabajo de Fin de Máster, y está compuesta por dos partes. En la primera parte está la página web en la que se puede hacer la edición de los materiales, explicada en el apartado 2.2.3.2, y en la segunda, un servicio web de traducción. En este apartado nos centraremos en la segunda parte.

Este servicio web de traducción *texto-picto*, permite dado un texto en español traducirlo a pictogramas. Esta traducción es la que, posteriormente, se utilizará en la página web de PICTAR. Este proceso de traducción de texto a pictogramas se ha implementado como un servicio web REST. Cuando el servicio web recibe el texto a traducir, lo primero que hace es un análisis

morfológico del mismo. De esta forma se obtienen las categorías gramaticales o lemas de cada una de las palabras.

Para este análisis se recurre a la herramienta Spacy, la cual comentaremos en apartados posteriores. Este proceso es necesario ya que en la base de datos de ARASAAC existen varios pictogramas que no están representados por una sola palabra sino por varias formando una expresión. Por ejemplo, existe un pictograma para la expresión "saltar a la pata coja", como podemos ver en la Figura 3.3. Sin este proceso, se traduciría a pictogramas cada una de las palabras de la expresión anterior. Esto podría suponer una dificultad para el lector ya que se perdería parte de la información que se quiere transmitir.

Una vez hecho el proceso de traducción, el servicio web PICTAR devolverá un JSON con las palabras que incluye el texto, agrupadas como n-gramas, junto con el o los identificadores de los pictogramas ARASAAC.

Figura 3.3: Pictograma ARASAAC "saltar a la pata coja"

3.2.3.2. Google Cloud Vision API

La API de Google Cloud Vision[2] es una herramienta proporcionada por Google. Esta API proporciona una interfaz REST que contiene potentes modelos de aprendizaje automático que les permite a los desarrolladores de aplicaciones poder analizar una gran cantidad de imágenes, extrayendo así la información de las mismas para comprender su contenido. Google Cloud Vision ofrece algunas funciones como las que veremos a continuación:

- Detección de rostros: esta herramienta tiene como objetivo detectar rostros humanos contenidos en las imágenes, además intenta detectar atributos faciales como el estado emocional, etc. Para llevar a cabo este reconocimiento facial se espera que coincidan la información biométrica del rostro detectado con la información biométrica de caras que tengan almacenadas en la base de datos.
- Detección de puntos de referencia: tiene como objetivo reconocer los edificios o estructuras naturales dentro de una imagen e identificar su ubicación (latitud y longitud) en el mapa.

[2] Para más información acceder a `https://cloud.google.com/vision/`

- Detección de logotipos: detecta los logos de las marcas dentro de una imagen.
- Detección de búsqueda segura: esta función es desarrollada por *Google SafeSearch* y detecta cualquier contenido inapropiado dentro de la imágenes, tales como contenido dirigido a adultos o los de carácter violento.
- Detección de textos: de una imagen se extrae el texto contenido en la misma mediante la técnica OCR, que explicaremos en el siguiente apartado. Esta función, además, ofrece la identificación automática del idioma. Por la orientación del presente trabajo nos centraremos en el uso de esta funcionalidad en particular.

3.3. Reconocimiento Óptico de Caracteres (OCR)

Cada vez que leemos una revista, un libro o incluso este trabajo de fin de grado, sin darnos cuenta nuestros ojos están reconociendo los caracteres que están presentes en los textos, mientras que nuestro cerebro procesa estos caracteres para entender el significado de las letras, palabras o el contenido del texto en sí. No parece imposible pensar que, con el continuo avance de la tecnología, un computador pueda hacer lo mismo o al menos un intento de ello.

Cuando hablamos de Reconocimiento Óptico de Caracteres, generalmente conocido como OCR por sus siglas en inglés *Optical Character Recognition*, hablamos de una tecnología que intenta imitar la facultad que tiene el ojo para percibir y reconocer cada elemento que le rodea. OCR es un proceso que nos permite convertir textos contenidos en imágenes en textos editables. Estas imágenes pueden ser desde documentos escaneados hasta imágenes tomadas con una cámara digital.

3.3.1. ¿Cómo funciona el OCR?

El principal objetivo (Sandín Carral, 2015) de los algoritmos de OCR es distinguir y reconocer los caracteres de un texto dentro de una imagen cualquiera. Sin embargo, esta no es una tarea trivial.

Imaginemos por un momento cuántas formas hay de escribir la letra *H*, cuántos estilos y tipos de letras (fuentes) existen actualmente. En la Figura 3.4 podemos ver algunos ejemplos de los mismos. Pues bien, aunque veamos esta letra escrita con diferentes formas y estilos, sabremos distinguirla debido a que sigue un único patrón que la distingue del resto de caracteres: dos líneas verticales paralelas y una línea horizontal que las une por la mitad.

Figura 3.4: Diferentes fuentes de la letra *H*

El proceso de reconocimiento óptico de caracteres se realiza en cuatro etapas:

- *Pre-procesamiento*: en esta primera etapa se ejecutan una serie de técnicas para intentar que la imagen quede lo más clara posible. Algunas de las técnicas son:
  - Binarización: si la imagen escaneada está a color o en escala de grises se convertirá a blanco y negro. De esta forma se distinguirán los caracteres del resto de la imagen.
  - Alineación: si la imagen queda torcida tras el escaneo, se alinearán los caracteres.
  - Limpieza (*Despeckle*): se eliminarán las manchas de la imagen y los bordes suavizados.
- *Segmentación de la imagen*: es el proceso más costoso de todas las etapas del reconocimiento de caracteres. La segmentación se basa en la detección de caracteres utilizando una serie de procedimientos para identificar contornos y delimitar regiones de la imagen. Debido a que cada carácter es distinto en cuanto a forma y tamaño, no hay un método general que facilite la implementación de esta etapa.
- *Reconocimiento del caracter*: podría decirse que es la etapa más importante en este proceso, y OCR utiliza dos formas distintas para este reconocimiento.
  - Reconocimiento de patrones: Este método se encarga de reconocer el caracter completo comparándolo con otros caracteres (patrones) almacenados en su base de datos. El inconveniente de este método es que para que sea efectiva la comparación entre caracteres, tanto el caracter que se intenta reconocer como su patrón almacenado en la base de datos tendrán que ser de la misma fuente. Actualmente, este método se realiza aplicando Redes Neuronales.

Esto aporta una gran ventaja ya que las redes neuronales tienen un buen desempeño a la hora de aprender a reconocer formas y patrones.

- Detección de características: Este método es más sofisticado en cuanto a la detección de caracteres. También es conocido como *extracción*, ya que su función principal es "separar" las características de los caracteres como líneas, curvas, posiciones de las líneas, ángulos e intersecciones, entre otras. Retomando el ejemplo de la letra *H*, observamos que se puede descomponer en tres líneas: dos verticales y una horizontal, como se muestra en la Figura 3.5. Una vez detectada la característica que distingue al caracter *H* del resto, será mucho más fácil reconocer cualquier letra independientemente de su fuente.

- *Post-procesamiento*: esta última etapa se lleva a cabo después de haber hecho el reconocimiento de caracteres, realizando una corrección básica de errores. Algunos *software* OCR incluyen un corrector ortográfico que detecta palabras mal escritas. Otros programas más sofisticados ofrecen funcionalidades extra para detectar y corregir errores. Por ejemplo, algunos programas utilizan el análisis de palabras adyacentes para encontrar conjuntos de palabras que suelen aparecer juntas. De esta forma, un texto mal reconocido como "el berro ladra" podría ser automáticamente corregido a "el perro ladra", ya que "perro" y "ladra" son palabras que suelen aparecer juntas.

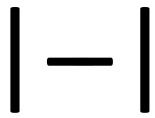


Figura 3.5: Extracción de las características de la letra *H*

3.3.2. ¿Por qué es útil el OCR?

Actualmente, el uso del reconocimiento óptico de caracteres nos ofrece muchas ventajas. Son innumerables las horas de trabajo que se podrían haber ahorrado en las conversiones, a mano, de los textos contenidos en imágenes o de documentos físicos a texto digital. Además, el hecho de poder automatizar este proceso de conversión hace que se reduzca el número de errores (humanos) que se pueden cometer en su realización.

Desde un punto de vista práctico, el uso del OCR ha hecho posible la transformación y digitalización de libros o documentos históricos que, actualmente, solo existen en un formato físico. Otro de sus posibles usos sería

ordenar, de alguna forma, las imágenes de documentos en una base de datos. Por ejemplo, si tenemos un conjunto de documentos escaneados, podría ser interesante ser capaz de obtener los datos contenidos en dichos documentos. Si se conoce que todas estas imágenes tienen algún campo en común, podría utilizarse para hacer búsquedas relativas a dicho campo. Otra de las utilidades, que será la que se aplique al presente trabajo, es la de combinar el OCR con otras técnicas como el procesamiento del lenguaje para llevar a cabo la traducción de textos a lectura fácil o pictogramas y hacer posible que personas con algún tipo de discapacidad cognitiva o deficiencia visual o auditiva tengan acceso a la información.

3.3.3. Proyectos relacionados: Google Traductor

Google Traductor, o *Google Translate* en inglés (IIMED, s.f.), es una de las aplicaciones móviles más conocidas en la actualidad. Es capaz de traducir el texto que queramos a cualquiera de los 103 idiomas que dispone la aplicación. Fue creada en 2006 por Google y progresivamente se han ido agregando más funcionalidades. Actualmente Google Traductor nos permite hacer una traducción desde un texto escrito, un mensaje de voz, una foto con texto (como se muestra en la Figura 3.6) o incluso en tiempo real, con la cámara del móvil. Para estas dos últimas funcionalidades Google Traductor incorpora la tecnología OCR.

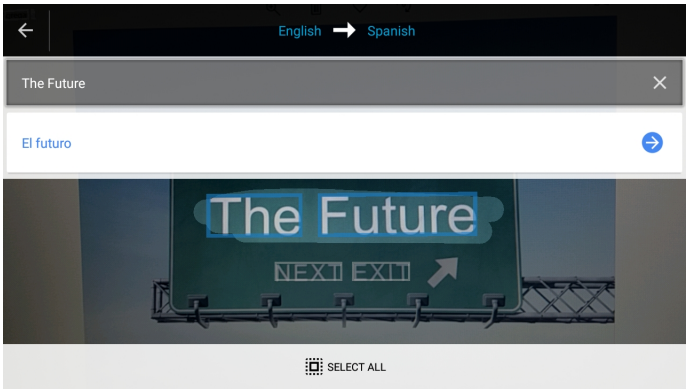


Figura 3.6: Ejemplo de traducción de texto usando la aplicación Google Traductor

3.4. Procesamiento de Lenguaje Natural (PLN)

El Procesamiento de Lenguaje Natural (Villena Román et al., 2011), también conocido como NLP por sus siglas en inglés *Natural Language Pro-*

cessing, es una rama de la Inteligencia Artificial que lleva más de 50 años en desarrollo e investigación. El PLN intenta conseguir que un ordenador entienda, interprete y manipule el lenguaje natural como el español, el inglés o el chino, entre otros, para lograr una comunicación con el ser humano.

3.4.1. Estructura del Procesamiento de Lenguaje Natural

Para lograr el objetivo del procesamiento del lenguaje natural es necesario que la máquina comprenda la estructura de dicho lenguaje. Dicho procesamiento tiene en cuenta, entre otros, los siguientes aspectos:

- Fonología: relaciona las palabras con los sonidos que representan.
- Morfología: forma palabras a partir de los morfemas. A nivel morfológico una misma palabra puede tomar diferentes roles morfosintácticos en función del contexto en el que aparece. Esto puede ocasionar problemas de ambigüedad. Por ejemplo, en la frase "Deja los sobres que sobren sobre la mesa", la palabra *sobre* es ambigua morfológicamente.
- Sintaxis: se encarga de las estructuras de las frases, es decir, como se relacionan todas las palabras de una frase entre sí, su organización y cual o cuales son las palabras más importantes. El objetivo de este nivel es asignar etiquetas a cada una de las palabras dentro de la frase. Estas etiquetas sirven para diferenciar entre un adjetivo, un adverbio, un sustantivo, etc. De esta forma se sabrá cómo se combinan las palabras y si forman estructuras gramaticales correctas.
- Semántica: se estudia el significado de las palabras o frases. El análisis semántico incluye el uso de la semántica léxica y de la semántica oracional. La primera se refiere al estudio del significado de las palabras de forma individual, mientras que la segunda estudia cómo se combinan estas palabras para formar significados complejos. El significado de una frase no se basa en el significado individual de las palabras que la componen, sino en la relación que tienen estas palabras entre sí.

3.4.2. Herramientas para el Procesamiento de Lenguaje Natural

Dado que el Procesamiento del Lenguaje Natural es una disciplina muy compleja y se escapa del alcance de este trabajo, es necesario la utilización de herramientas ya desarrolladas para poder realizar dicho análisis. A continuación, vamos a explicar algunas de estas herramientas de procesamiento de lenguaje natural que existen actualmente.

3.4.2.1. OpenNLP

OpenNLP[3] es una librería de Apache escrita en Java que comprende un conjunto de funcionalidades basadas en el aprendizaje automático para el procesamiento de texto en lenguaje natural. Algunas de las tareas de procesamiento de lenguaje natural que soporta esta librería son:

- *Tokenization*: dada una secuencia de caracteres, la *tokenización* es la tarea de cortarla en trozos, llamados *tokens*, que serán cada una de las palabras que componen la frase. Además, en este proceso se pueden eliminar ciertos caracteres especiales como el punto, la coma, signos de exclamación e interrogación, entre otros.
- *POS tagging* (etiquetado gramatical): es el proceso de asignar o etiquetar a cada una de las palabras de un texto su categoría gramatical.
- Reconocimiento de entidades nombradas: se encarga de extraer y clasificar conjuntos de palabras en categorías predefinidas como personas, lugares, expresiones de tiempo y organizaciones, entre otras.
- *Chunking*: es el proceso de extraer secuencias de palabras que, en conjunto, tengan un significado. Por ejemplo, si tenemos la frase *La crema solar*, en lugar de separar la frase en *tokens*, se separaría por grupo de palabras *"la"* y *"crema solar"*, de esta forma no se perdería el significado de la frase.
- Lematización: es un proceso que consiste en hallar el lema correspondiente de una palabra flexionada, es decir, una palabra en plural, conjugada, en femenino, etc. Se llama lema a la palabra que se considera como representante de todas las palabra flexionadas. Por ejemplo, el lema *gat-* representaría a las palabras *gatas* o *gatito*.

En un primer momento pensamos en usar OpenNLP ya que está desarrollada en Java. Sin embargo, ha sido descartada porque no cumplía con los requisitos de nuestra aplicación, ya que usamos el idioma español y aunque esta herramienta lo soporta no funciona suficientemente bien.

3.4.2.2. SpaCy

SpaCy[4] es una librería *open source* para el procesamiento de lenguaje natural escrita en Python y Cython. Surgió en el 2015 como una herramienta

[3] Para más información consultar el manual de OpenNLP: `https://opennlp.apache.org/docs/1.9.1/manual/opennlp.html`

[4] Para más información acceder a `https://spacy.io/`

potente y a día de hoy es considerada una de las mejores en esta rama de la Inteligencia Artificial. Es una de las herramientas más rápida que existen ya que cuenta con modelos de redes neuronales convolucionales. Esta herramienta, a diferencias de otras, proporciona soporte para un total de ocho idiomas distintos, incluyendo el español. En este caso, el procesamiento de lenguaje español funciona mejor que otras herramientas de PLN.

Para la implementación de nuestra aplicación utilizaremos el servicio web API PLN NIL, basado en SpaCy, del que hablaremos en el capítulo 5.

Capítulo 4

Diseño de la aplicación

"Hay dos formas de realizar el diseño de una aplicación: la primera es el hacerlo tan sencillo que sea obvio para todos que no tiene deficiencias y la segunda es el hacerlo tan complicado que no queden deficiencias obvias"
— Tony Hoare

En el presente capítulo nos disponemos a hablar sobre el proceso de diseño de la aplicación, las dificultades que nos hemos ido encontrando en los distintos modelos que hemos ido desarrollando y los cambios que hemos realizado hasta alcanzar la versión final del proyecto.

4.1. Introducción

La idea con la que se lleva a cabo la implementación de este modelo de aplicación es facilitar la comprensión de cualquier tipo de texto que se puedan encontrar, en su día a día, aquellas personas las cuales presenten algún tipo de discapacidad cognitiva que les dificulta la comprensión lectora.

A su vez queríamos que fuese una herramienta que el usuario tuviera a su alcance en cualquier momento y que contara con aquellas funcionalidades que le resultasen más útiles. Con esto en mente, aparecían tres puntos a tener en cuenta: diseño, funcionalidades y entorno operativo.

4.1.1. Diseño y modelado

Con el objetivo de conseguir un diseño y modelado apropiado para la aplicación, tuvimos que ponernos en el lugar de aquellos usuarios para los

que está orientado el proyecto. Para ello estructuramos el plan de trabajo de la siguiente manera:

- Modelado de mockups de manera competitiva. Realizamos dos modelos iniciales de lo que sería la aplicación de manera individual. De esta manera teníamos dos puntos de vista distintos de como enfocarlo y podíamos combinar ambos para llegar a la creación final de la aplicación.
- Reuniones con nuestras tutoras del proyecto. Para que todo este trabajo diese sus frutos, nos apoyamos en el conocimiento previo que ya tenían nuestras tutoras Raquel Hervás y Susana Bautista, quienes ya han trabajado con anterioridad en otros proyectos similares. La oportunidad de contar con este apoyo nos permitió ajustar el diseño lo máximo posible a las necesidades que se pretenden cubrir, desde un punto realista y, que favorezca al usuario desenvolverse con comodidad.
- Reuniones con los expertos. Finalmente, tras tener un modelado final de lo que pensábamos que podía ser la aplicación, nos reunimos con una serie de expertos del Colegio de Educación Especial Estudio3 AFANIAS quienes están acostumbrados a trabajar con grupos de niños que presentan discapacidades cognitivas para las que está orientada nuestra aplicación. Con este aporte más cercano al usuario final, pudimos tomar una mejor referencia con respecto al correcto modelado de la herramienta.

Una vez teníamos la idea tomada, nos enfocamos en los principales elementos visuales de la aplicación como podían ser:

- La presentación del texto capturado en la interfaz de la aplicación.
- Como mostrar la respuesta de los servicios web, de los que hace uso la aplicación.
- La disposición y tamaño de los botones en la pantalla.

En definitiva, plantearnos el reto de conseguir una interfaz fluida y amigable con el usuario y que fuese personalizable para cada usuario que vaya a hacer uso de la aplicación.

4.1.2. Funcionalidades de la aplicación

Las funcionalidades que pensamos que deberían estar presentes en la aplicación son aquellas que se han pensado que podían ser más útiles para usuario y frente a las pequeñas dificultades que se pueda encontrar a la hora de entender algún texto.

Estas funcionalidades son las siguientes:

- Sinónimos: el objetivo de esta funcionalidad dentro de la aplicación es que, cuando un usuario se encuentre ante un texto en el cual desconozca el significado de una o varias palabras del mismo, se vea ante la posibilidad de encontrar otras similares y de esta forma intentar solventar el problema.
- Antónimos: el objetivo de incorporar esta funcionalidad se trata de la posibilidad de que, al igual que en el caso anterior, el usuario tenga la posibilidad de encontrar palabras opuestas a la previamente elegida. El contexto de este servicio es más limitado, pero en caso de un estudiante que tenga que hacer algún ejercicio de este tipo, encontramos interesante que la aplicación lo contenga.
- Definiciones: la presencia de esta funcionalidad es otorgar al usuario la posibilidad de que, en caso de que exista una palabra en un texto de la cual desconozcá totalmente su significado, tenga un conjunto de definiciones que le permitan situar que el significado de la palabra en el contexto de lo que está leyendo.
- Pictogramas: en muchas ocasiones la capacidad del usuario para entender una palabra o un texto puede ser más limitada, y una forma de simplificarlo es haciendo uso de los pictogramas, es decir, ilustraciones mediante las cuales el usuario consigue entender el significado de una palabra o comprender el contexto de lo que está leyendo.
- Resumen: en ciertas ocasiones el usuario puede encontrarse con textos cuya complejidad gramatical sea muy alta, para estas ocasiones contamos con esta funcionalidad. Lo que ofrecemos con esta opción al usuario, es la posibilidad de seleccionar un texto completo y que se le devuelva un resumen del mismo donde se extrae la idea principal.

En todos los casos el usuario podrá hacer una fotografía de un texto con su móvil, y dentro de él usar cualquiera de estas funciones para entenderlo mejor.

4.1.3. Entorno operativo

El entorno elegido para el desarrollo de la aplicación se trata de Android. La principal razón por la que se ha elegido este entorno es porque como se comenta desde un principio, intentamos desarrollar una herramienta disponible en el día a día de la persona y que pueda hacer uso de esta en cualquier momento.

Ventajas de una aplicación en Android:

- Volumen de usuarios al que se puede llegar. Hoy en día todo el mundo lleva un móvil consigo a la hora de ir a trabajar, estudiar, etc. Además, que el entorno elegido sea Android es porque se trata del entorno operativo móvil más globalizado y con el que nos va a ser más fácil llegar al usuario.
- Siempre está disponible para su uso. Al tener la aplicación instalada directamente en el dispositivo móvil, es muy cómodo para el usuario llegar a utilizarla en caso de que le surja la necesidad, pues se trata de abrir un dispositivo que siempre lleva consigo.
- Experiencia del usuario. La posibilidad de crear un entorno totalmente personalizado a las necesidades del usuario, es decir, que sea él mismo el que decida qué funciones quiere tener operativas a la hora de usar la aplicación, darnos la oportunidad de realizar un diseño optimizado, etc.
- La posibilidad de dar una herramienta nueva y útil en el día a día.

4.2. Propuestas de diseño

Como exponemos en los puntos anteriores, la principal preocupación a la hora de desarrollar el proyecto es conseguir una interfaz que no solamente sea amigable con el usuario, sino que sea personalizable. Como no todos los usuarios van a tener el mismo grado/tipo de discapacidad cognitiva hay que conseguir que esta se adapte a las necesidades del usuario que vaya a utilizarla. Por tanto, los puntos que hemos tenido en cuenta a la hora de realizar el diseño son:

- Debe de tratarse de un diseño simple, donde la información mostrada sea clara y concisa.
- Los elementos como los botones tienen que tener un tamaño significativo. De esta manera facilitamos el uso de la aplicación al usuario.
- Una vez que hemos mostrado el resultado de la funcionalidad elegida por el usuario, la idea es mantener el texto original junto con el propio resultado. De esta manera permitimos que el usuario tenga la oportunidad de realizar una lectura fluida y de fácil comprensión, al tener junto con el texto la respuesta a aquella duda que le surge.
- La interfaz de la aplicación no se tratará de un modelo fijo, sino que será adaptable a las necesidades de los usuarios. El usuario podrá elegir aquella funcionalidades que desea tener operativas dentro de la aplicación.

Con este planteamiento, comenzamos con la creación de los primeros modelos de mockups, que darán lugar a las primeras reuniones. De estas reuniones se sacarán los puntos a mejorar, llegando así al diseño final de la aplicación, el cual se ajustara a las necesidades planteadas.

4.3. Modelado de la interfaz mediante mockups

Se realizaron tres iteraciones en el diseño de la interfaz.

4.3.1. Primera iteración

En este punto contemplamos el primer modelo realizado, a través de una serie de mockups, mediante los cuales se pretendía mostrar la interacción básica que el usuario tendría con la aplicación a través de su dispositivo móvil.

Como hemos comentado inicialmente, se realizaron dos modelos de forma paralela e independiente, con el objetivo de tener una visión más global de lo que podía llegar a ser el diseño de la interfaz. A continuación, se mostrarán los modelos en sus distintas fases.

- Modelo 1. Como se observará a continuación, se trataba de un diseño básico, con una distribución de los elementos que pretendía ajustarse lo máximo posible al planteamiento expuesto.

Figura 4.1: Pantalla de inicio de la aplicación.

En este primer modelo, una vez iniciamos la aplicación como se muestra en la Figura 4.1, nos encontramos con una vista donde tendríamos

la ventana en la que se mostraría el texto capturado, así como los funcionalidades disponibles para el usuario (sinónimos, pictogramas, etc.).

A su vez, en esta vista encontramos el botón para acceder a la cámara para capturar el texto que queramos que sea analizado, así como una opción de menú despegable, donde como más adelante mostraremos, tendremos las opciones para ajustar la interfaz a los servicios que el usuario necesita.

Figura 4.2: Menú despegable y opción de ajustes de interfaz.

Como se puede observar en la Figura 4.2, dentro del menú desplegable de la esquina superior derecha encontramos un pequeño menú donde dispondremos de distintas opciones. Entre ellas encontraremos la posibilidad de activar el flash de la cámara de forma automática cada vez que accedamos a la cámara mediante la aplicación, así como tener activado el autoenfoque de la misma cada vez que se active la cámara.

Por otro lado, disponemos del menú que nos da acceso a la reorganización de la interfaz. Dicho menú nos permitirá ajustar la interfaz de la aplicación acorde con las funcionalidades que el usuario vaya a utilizar o en caso de que vaya a utilizar el dispositivo un usuario diferente al habitual, poder ajustar la aplicación para dicho usuario.

En la Figura 4.3 encontramos una de las propuestas realizadas a la hora de mostrar el resultado devuelto por la opción elegida por el usuario. La idea de mostrarlos justo debajo del texto se trataba de que de esta manera el usuario pudiera relacionarlo directamente con la parte del texto donde se había encontrado con dificultades para su correcta comprensión, permitiendo una interpretación más directa del significado de la frase/palabra seleccionada.

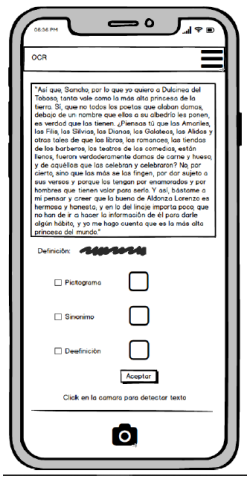


Figura 4.3: Modelo de vista de como se muestra el resultado del servicio seleccionado.

- Modelo 2. Esta versión de la interfaz se trata de un segundo modelo que realizamos junto con el anterior para la primera iteración. De esta manera, contábamos con dos puntos de vista diferentes de como enfocar el funcionamiento de la aplicación.

Figura 4.4: Pantalla tras haber tomado una foto del texto a procesar.

En la Figura 4.4 podemos observar como en este modelo de interfaz, el texto que el usuario quiere analizar es captado mediante una fotografía del cartel. La idea de hacerlo de esta manera era que se mantenía la referencia de donde se sacaba el texto.

Como observamos en la Figura 4.5, una vez elegimos el servicio de la aplicación que queremos utilizar, el texto se dividirá en palabras. De

Figura 4.5: División del texto de la imagen en palabras y selección de la palabra a procesar.

esta manera, el usuario podría marcar con suma facilidad cuál es la palabra que no entiende. Sin embargo, esto limitaba el uso de algunos servicios de la aplicación, como podría ser "lectura fácil/resumen", los cuales se encargan de traducir/resumir frases o texto complejos a otros más sencillos.

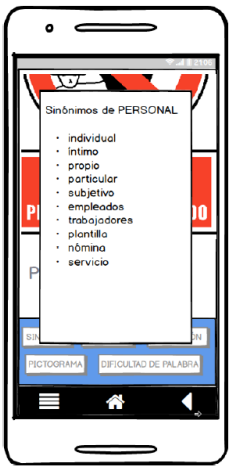


Figura 4.6: Muestra de cómo se refleja la respuesta del servicio web utilizado.

Por último, en la Figura 4.6, observamos que para mostrar el resultado, a diferencia de en el modelo 1, este se mostraría en una ventana emergente. Fue una idea que inicialmente tuvimos en cuenta, ya que permitía que, de una forma clara, el usuario tuviera la respuesta que buscaba. Sin embargo, podría suponer un problema a la hora que no se observa dentro del contexto del que procede la palabra/frase elegida.

Tras la presentación de esta primeras versiones, nos dimos cuenta de que había determinados aspectos que había que mejorar o adaptar para los usuarios hacía los que está orientada la aplicación. Algunas de las que había que llevar a cabo eran:

- La orientación de los botones, es decir, tamaño y disposición, así como el número de botones de servicio que se muestran por pantalla. Los problemas que observamos en estos aspectos eran que los botones debían de tener un mayor tamaño incluyendo un pictograma que indicara la funcionalidad a la que da acceso.
- La disposición de los botones y que el número de botones que se pudiesen mostrar en la ventana principal del teléfono es limitado, por lo que se llegó a la conclusión de presentar una "ventana deslizante" de tal forma, que en cada ventana se presenta un número máximo de cuatro botones, sin limitar el espacio establecido para el texto y la respuesta de los servicios.

4.3.2. Segunda iteración

En esta segunda vuelta se muestran dos modelos de mockups, los cuales son una evolución de los anteriormente presentados. En estos diseños se lleva a cabo la corrección de los errores, con respecto a los puntos a mejorar que pudimos encontrar en la revisión llevada a cabo sobre los diseños que se habían realizado anteriormente.

- Modificaciones en el modelo 1. En la Figura 4.7 podemos observar uno de los cambios realizados con respecto a la disposición de los botones, tamaño y pictogramas que permiten al usuario un manejo más cómodo y amigable de la aplicación. A su vez el número de funcionalidades disponibles no se ve limitado como en la versión anterior (ver Figura 4.1) gracias a la disposición de las "ventanas deslizantes".

  Uno de los puntos que supusimos que podía causar dificultades en el uso de la aplicación, era como se realizaba la selección de las frases/palabras sobre el texto capturado mediante la cámara. Originalmente, una vez capturado el texto, este se mostraba en la pantalla principal de la aplicación y el usuario debía pulsar sobre la parte del texto que quería que fuese analizada por alguno de las funcionalidades. Sin embargo, esto podía resultar incomodo y decidimos cambiarlo. Actualmente, una vez el texto es capturado a través de la cámara, este es procesado y dividido en frases (Figura 4.8) las cuales son seleccionables como si de un *radio button* se tratase. De esta forma el usuario solo tendrá que seleccionar la frase en cuestión y pulsar el botón de la funcionalidad a utilizar.

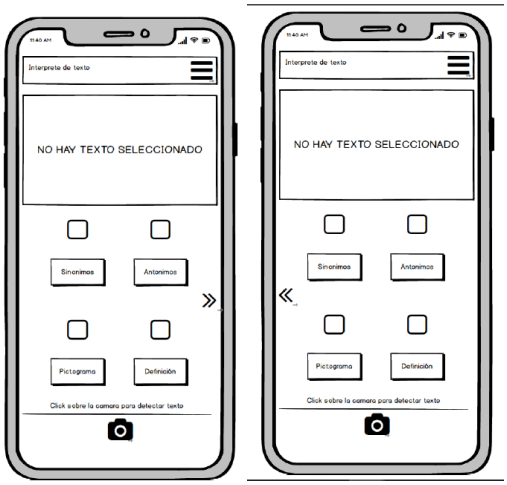


Figura 4.7: Nuevo modelo de pantalla inicio, ajustándose a los nuevos parámetros.

Como último punto a comentar y en relación con lo mencionado anteriormente, supusimos que en el caso de que el usuario quisiera seleccionar una palabra de una frase en concreto, puede resultar útil que una vez la frase sea seleccionada esta se divida en las palabras que la componen, (Figura 4.9). La idea de este funcionamiento es facilitar la interacción entre el usuario y el texto.

- Modificaciones en el modelo 2. En la Figura 4.10, observamos el proceso de selección del texto por parte del usuario, y como una vez elegido nos saldría la opción de cómo queremos que este sea presentado en la aplicación, si a nivel de frase o palabra.

  A continuación, basándonos en el caso de que el usuario decidiese que el texto fuese procesado por frases, este se presenta en la Figura 4.11. Vemos un modelo de interfaz amigable con el usuario, donde la interacción con el texto es sencilla y la manera de presentar la información es clara y vistosa.

  Por último podremos ver en la Figura 4.12, donde tendríamos uno de los formatos en los que se muestra como es la respuesta de la aplicación. El texto original y la solución devuelta por el servicio web utilizado estarían integradas ambos en la misma pantalla, lo que permite que no se descontextualice la respuesta que el usuario busca con respecto al texto del que proviene. Esto le permite una mejor comprensión y mayor fluidez.

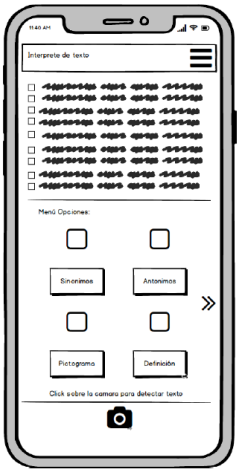


Figura 4.8: Modelo de pantalla, para la selección de una palabra en una frase concreta del texto.

4.3.3. Tercera iteración

Tras la revisión de los mockups expuestos en la segunda iteración, llegamos a la conclusión de que había algunos cambios que llevar a cabo.

- Modelo final. Como observamos en la Figura 4.13, uno de los cambios que vimos que había que realizar, con respecto a la versión anterior era que el texto dividido en frases no valía para los servicios de lectura fácil y resumen.

  Para resolver este problema, decidimos que cada vez que seleccionásemos uno de estos servicios, el texto dividido en frases volvería al formato original y en la pantalla del móvil se mostrarían tanto el texto original como el resumen del mismo.

4.4. Validación del diseño con los expertos

El Colegio de Educación Especial Estudio3 AFANIAS[1] es un centro de educación especial concertado con la Consejería de Educación, Juventud y Deporte de la comunidad de Madrid. En él se escolarizan alumnos de tres a veintiún años que presentan algún tipo de discapacidad intelectual, plurideficiencias o trastorno generalizado del desarrollo.

A mediados de marzo concertamos una primera reunión con el colegio para validar el diseño con los expertos. Acudieron algunos profesores del

[1] Para más información acceder a `http://colegioestudio3.blogspot.com`

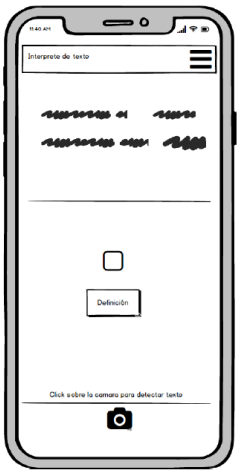


Figura 4.9: Nuevo modelo de pantalla inicio, ajustándose a los nuevos parámetros.

centro y parte del equipo directivo. En ella expusimos nuestras ideas sobre las distintas funcionalidades que podría tener la aplicación. Nos centramos en exponer, brevemente, su utilidad y en que ámbito se podría hacer uso de la misma. Enseñamos los dos últimos prototipos que teníamos, navegando entre las diferentes pantallas y mostrando las funcionalidades que ofrece dicha aplicación. Una vez finalizada la exposición, comenzaron las preguntas.

Por un lado, nos planteamos si los alumnos del colegio tendrían autonomía suficiente para manejar un dispositivo móvil. Los profesores nos comentaron que sí, y que incluso algunos de ellos interactuaban y se desenvolvían con aplicaciones que no estaban dirigidas a personas con discapacidad intelectual. Esto supuso un alivio para nosotros ya que hay algunas funcionalidades dentro de la aplicación, como la de deslizar la pantalla o configurar opciones, que no sabíamos si podrían suponer una barrera para ellos.

Por otro lado, los profesores del colegio nos hicieron recomendaciones de nuevas funcionalidades que podríamos añadir a la aplicación, de cara a cubrir un mayor rango de necesidades. Una pregunta que nos hicieron fue si la aplicación tenía alguna opción para leer el texto capturado. Hasta ahora no habíamos contemplado esta opción en ninguna de las versiones. Según los profesores, sería muy útil ya que hay alumnos que no tienen facilidad para la lectura, tanto para texto normal como para pictogramas, y tener esta opción les ayudaría a desenvolverse en cualquier entorno. También nos comentaron que muchos de los niños del colegio entienden mejor un texto escrito en mayúsculas que en minúsculas. Por esto, nos recomendaron añadir una funcionalidad para poder decidir si queremos ver el texto capturado en mayúsculas o minúsculas. Además, el tipo de letra que suelen usar en sus clases es *Arial*.

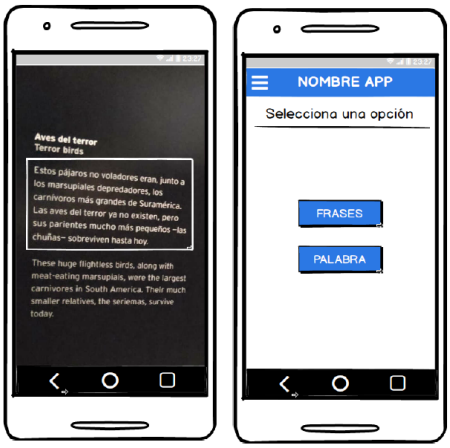


Figura 4.10: Captura del texto a analizar y toma de decisión del procesado de texto.

Respecto a las opciones que ya estaban en el diseño, les pareció interesante poder configurar las funcionalidades, es decir, que el propio usuario pueda escoger si quiere tener activo una funcionalidad u otra. Por ejemplo, si solo queremos usar la opción de traducción a pictogramas y no queremos acceder a ninguna otra como buscar definiciones, sinónimos, etc. O el hecho de poder dividir el texto en frases para seleccionar cuales queremos traducir a pictogramas o lectura fácil, por ejemplo.

4.4.1. Diseño definitivo de la interfaz

Una vez recogida toda la información y recomendaciones de los profesores, hicimos modificaciones de la interfaz y añadimos las nuevas funcionalidades. En la Figura 4.14a se muestra la pantalla después de haber capturado el texto. Se han añadido dos nuevos botones debajo del cuadro donde se muestra el texto capturado. En la parte izquierda, aparece un botón representado por el icono de audio. Al pulsar este botón la aplicación leerá en voz alta el contenido del texto. El botón de la derecha, representado por una *A* mayúscula y una *a* minúscula, sirve para cambiar la visualización del texto. Esto significa que, si el texto se muestra en minúsculas, al pulsar el botón cambiará a mayúsculas, y viceversa. Podemos ver un ejemplo en la Figura 4.14b. Estos dos servicios no son configurables, es decir, siempre estará activo para cuando el usuario quiera pulsarlo. El resto de los servicios de la aplicación permanecerán iguales que en las versiones anteriores.

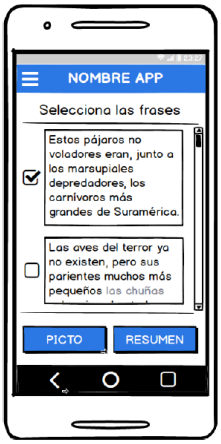
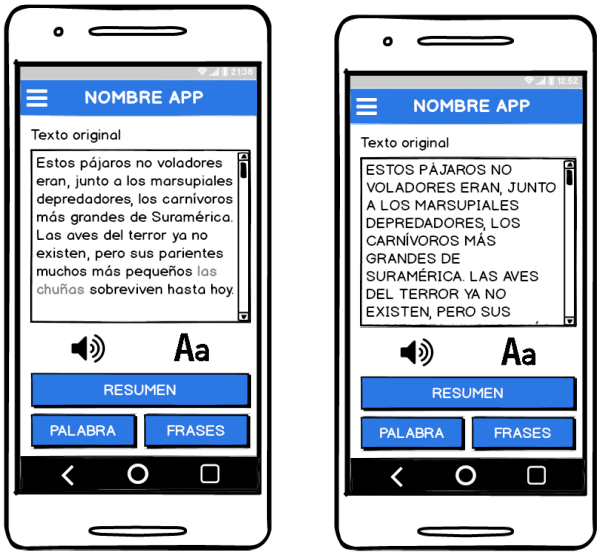


Figura 4.11: Selección sobre el texto dividido en frases.

Figura 4.12: Traducción de la frase seleccionada a Pictogramas.

Figura 4.13: Cambio de la visualización del texto capturado, una vez seleccionamos los servicios resumen/lectura fácil.



(a) (b)

Figura 4.14: Visualización del texto. Paso de minúsculas a mayúsculas

Capítulo 5

Implementación

"Lo que funciona bien es mejor a lo que se ve bien,
porque lo que funciona bien
permanece en el tiempo"
— Ray Eames

En este capítulo se realizará un recorrido por todas las pantallas de nuestra aplicación y se explicará en detalle todas las funcionalidades que ofrece. Además, hablaremos sobre cómo está implementada, así como los diversos servicios web que se utilizan para lograr el funcionamiento de la misma.

5.1. Descripción de la aplicación

A medida que íbamos desarrollando la interfaz de la aplicación, nos dimos cuenta de que algunos botones que habíamos diseñado en los mockups ocupaban demasiado espacio en la pantalla. Aunque la idea principal fue crear botones grandes para facilitar el uso de la aplicación al usuario final, tuvimos que reorganizar algunas ideas y modificar la interfaz. Por otra parte, propusimos varios nombres para la aplicación y escogimos *LeeFácil* y como logo para la misma tenemos el que se muestra en la Figura 5.1.

LeeFácil es una aplicación nativa desarrollada específicamente para dispositivos que tengan instalado el sistema operativo Android. Para poder hacer uso de la aplicación en nuestro dispositivo necesitamos tener una versión igual superior a Android 4.0.3 *IceCreamSandwich*. Además, se deberán conceder permisos para usar la cámara. Para facilitar su difusión hemos subido la herramienta a *Google Play Store*. La aplicación está disponible en el enlace: `https://play.google.com/store/apps/details?id=com.ucm.fdi.leefacil&hl=es`

Figura 5.1: Logo de la aplicación

5.1.1. Funcionalidades

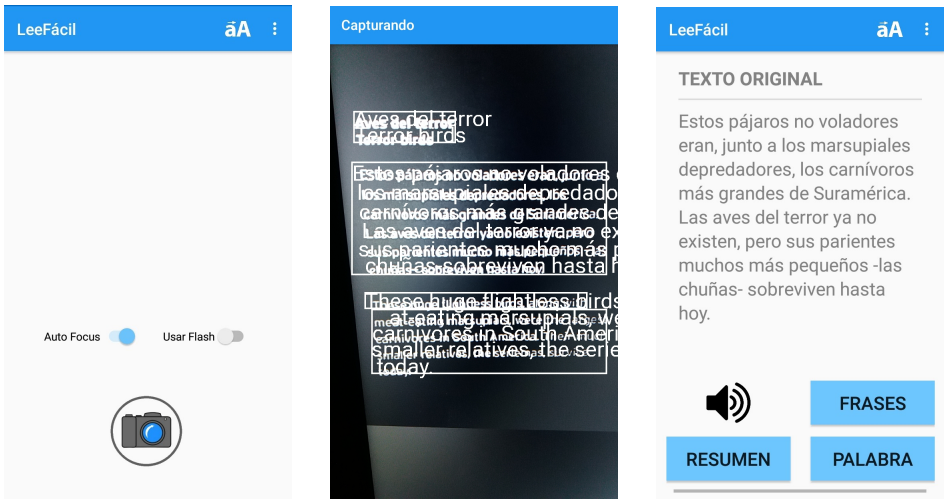
Al abrir la aplicación se muestra la pantalla principal (ver Figura 5.2a). En la parte superior de la pantalla nos encontramos con una barra. A la izquierda de la misma se muestra el nombre de la aplicación, *LeeFácil*, y a la derecha, aparecen las siguientes opciones:

- Botón (**aA**): al pulsar este botón, el texto capturado por el usuario cambiará de minúsculas a mayúsculas o viceversa.
- Botón (⋮): al pulsar este botón aparecen dos opciones.
  - AJUSTES: si pulsamos esta opción se mostrará una ventana en la que podemos seleccionar qué servicios queremos utilizar: *Definiciones*, *Sinónimos*, *Antónimos* o *Traducción a pictogramas*. Por defecto estarán marcados todos los servicios.
  - SOBRE *LeeFácil*: al pulsar esta opción aparecerá una ventana con la información referente a la aplicación.

Esta barra permanecerá activa en la mayoría de las vistas de la aplicación, lo que significa que podremos acceder a estas opciones desde cualquiera de ellas. Aún en la Figura 5.2a vemos que en la parte inferior aparecen dos opciones más, *Auto Focus* y *Usar Flash*, que van relacionadas con la cámara: la primera es para el autoenfoque, la cual permite enfocar en todo momento, y la segunda para el flash. Por último, aparece un botón representado por el pictograma ARASAAC *cámara*. Si pulsamos este botón nos llevará a la siguiente pantalla (ver la Figura 5.2b).

En esta pantalla se hará uso de la cámara del dispositivo. El usuario tendrá que apuntar la cámara al texto que desea capturar y pulsar el recuadro que se crea sobre el mismo. Una vez seleccionado el texto, tendremos disponibles las opciones principales que ofrece la aplicación. Podemos ver estas opciones en la Figura 5.2c.

A continuación, explicaremos detalladamente cada una de estas opciones.



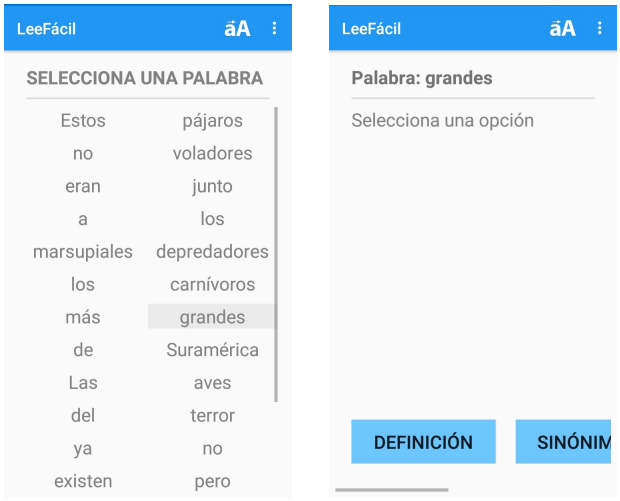
(a) (b) (c)

Figura 5.2: Tres primeras vistas de la aplicación

5.1.1.1. Palabra

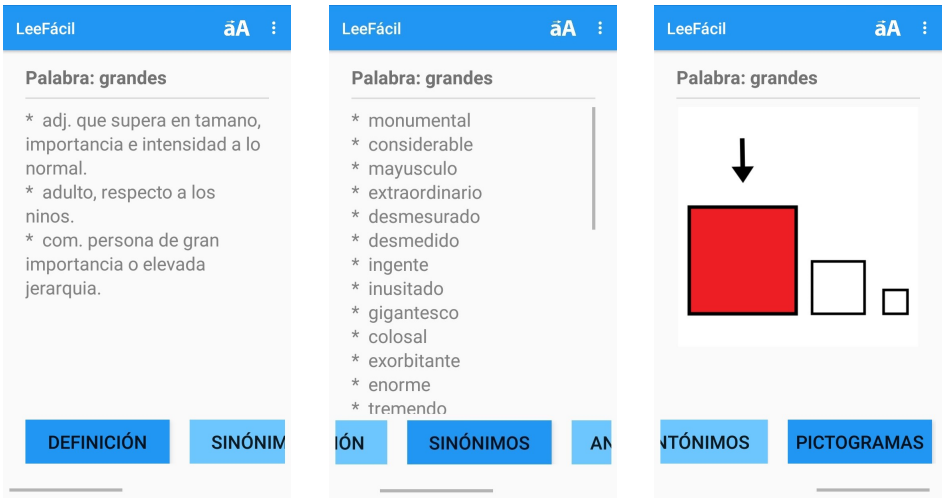
Al pulsar el botón *Palabra* se cambiará a una vista donde aparecerá el texto y podremos seleccionar una palabra dentro del mismo. Como se muestra en la Figura 5.3a, si seleccionamos la palabra *grandes* aparecerá otra vista con las opciones disponibles (ver Figura 5.3b). Estas opciones aparecen en la parte inferior de la pantalla y son las que podremos configurar. Si queremos verlas todas tendremos que deslizar la parte inferior de la vista avanzando por el *scroll*. Las opciones que ofrece la aplicación son las siguientes:

- *Definiciones*: muestra todas las acepciones de la palabra seleccionada. (ver Figura 5.4a)
- *Sinónimos* y *Antónimos*: muestra, si los tiene, todos los sinónimos y antónimos de la palabra, respectivamente (ver opción sinónimos en la Figura 5.4b). En caso de que la palabra no tenga sinónimos ni antónimos aparecerá un mensaje que anuncia que no se ha podido encontrar un resultado.
- *Traducción a Pictograma*: muestra la traducción a pictograma de la palabra. Si no existe el pictograma relacionado con esa palabra aparecerá una imagen con una cruz roja. En caso de que la palabra tenga más de un pictograma asociado, se podrá pulsar en la imagen y cambiará el *picto* (ver Figura 5.4c).



(a) (b)

Figura 5.3: Opción *Palabra*



(a) (b) (c)

Figura 5.4: Algunas funcionalidades dentro de la opción *Palabra*

5.1.1.2. Frases

Volviendo a la vista de la Figura 5.2c, si pulsamos el botón *Frases* nos llevará a una pantalla donde se mostrará el texto dividido por frases. Como se muestra en el ejemplo de la Figura 5.5a, podemos seleccionar una o varias de ellas y pulsar en los botones que aparecen en la parte baja de la vista.

A la derecha vemos que aparece un botón *Palabra* que corresponde con el botón *LF* (Lectura Fácil) que se diseñó en los *mockups*. A medida que íbamos desarrollando la aplicación nos dimos cuenta de que no existe, aún, una herramienta que adapte automáticamente los textos en castellano a Lectura Fácil, por lo que esta idea ha sido descartada en nuestro proyecto. Por tanto, si pulsamos el botón *Palabra*, se realizará todo el proceso comentado en el punto anterior, pero en vez de realizar el análisis con el texto capturado, se realizará con la frase o las frases seleccionadas.

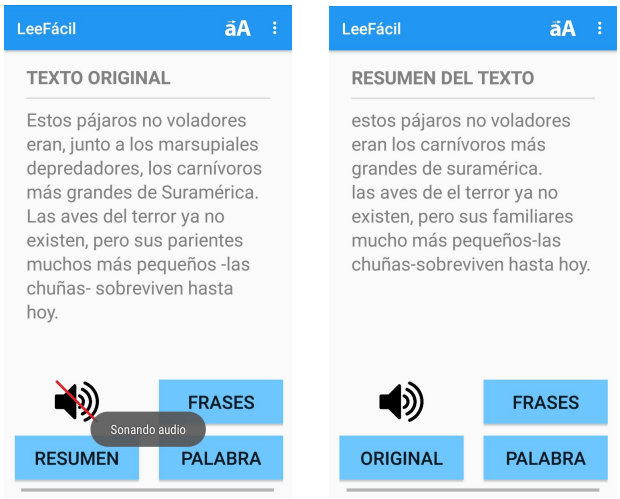
Por otra parte, si pulsamos el botón *Pictos* nos llevará a una vista en la que se mostrará la traducción de la frase a pictogramas (ver Figura 5.5b). De forma análoga a la opción de *Traducción a Pictograma* en *Palabra*, en caso de que la palabra tenga más de un pictograma asociado, se podrá pulsar sobre la imagen y esta cambiará de *picto*.

(a) (b)

Figura 5.5: Opción *Frases*

5.1.1.3. Lectura en voz alta

De nuevo en la Figura 5.2c podemos observar que existe, además, un botón representado por el icono de audio. Si lo pulsamos, el sistema leerá en



(a) (b)

Figura 5.6: Opciones *Lectura en voz alta* (a) y *Resumen* (b)

voz alta el texto que sale en la pantalla. Mientras esté activa esta opción, el botón aparecerá tachado. Si volvemos a pulsar, el sistema dejará de leer. En la Figura 5.6a podemos ver un ejemplo.

5.1.1.4. Resumen

Por último, si pulsamos el botón *Resumen* en la vista de la Figura 5.2c, nos llevará a otra pantalla en la que se verá un resumen del texto. En esta misma pantalla vuelven a aparecer todas las opciones comentadas anteriormente. Si desde este punto pulsamos algunas de esas opciones, el sistema trabajará con el texto resumido. Mostramos un ejemplo en la Figura 5.6b.

5.2. Arquitectura de la aplicación

LeeFácil ha sido desarrollada en Android Studio. Una de las ventajas de utilizar este IDE es que se creará una aplicación nativa. Esto significa que se aprovecharán más las funcionalidades del dispositivo, pero con el inconveniente de que su uso está limitado a dispositivos que tengan instalado el sistema operativo Android. Android Studio se basa en el lenguaje de programación Java para la parte lógica y en XML para la parte gráfica y, además, se hace uso de JSON para el intercambio de información entre la aplicación y los servicios web.

Dentro de Android Studio aparece el fichero `AndroidManifest.xml`, don-

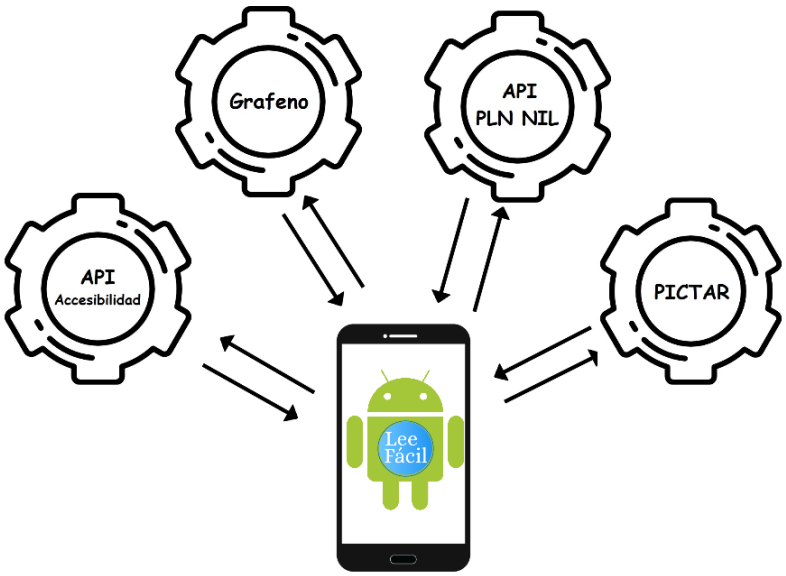


Figura 5.7: Arquitectura de *LeeFácil*

de están definidas todas las clases, vistas y permisos que son necesarios para usar la aplicación. Existen dos directorios importantes a mencionar: uno de ellos es el directorio `java` donde se almacenan todos los ficheros Java que llevarán la lógica de la aplicación y, por otro lado, tenemos el directorio `res` donde podemos encontrar la parte gráfica de la aplicación: vistas, menús, imágenes, ficheros de texto, entre otras.

En este proyecto se sigue una arquitectura *cliente-servidor*, es decir, existe un cliente (nuestra aplicación) que hace peticiones a los servidores, y estos servidores dan una respuesta. En la Figura 5.7 podemos ver un ejemplo de cómo está diseñada la arquitectura de la aplicación.

En esta sección, hablaremos sobre los servicios web externos a los que la herramienta se conecta. Además, explicaremos algunos aspectos internos, como clases implementadas o librerías utilizadas. Podemos encontrar todo el código de la aplicación en el repositorio: `https://github.com/NILGroup/TFG-1819-Aumentada`.

5.2.1. Configuración de funcionalidades

Dado que una de las ideas principales de la herramienta es ofrecer al usuario la posibilidad de escoger qué funcionalidades quiere usar, era necesario buscar una forma de almacenar datos. Estos datos tenían que guardarse en nuestra aplicación aunque se cerrara o incluso se apagara el dispositivo, ya que se entiende que el usuario no cambiará las funcionalidades que desea

utilizar. La idea era guardar cuatro variables que representaran si estaban activadas o no las opciones de *Definición*, *Sinónimos*, *Antónimos* y *Traducción a pictogramas*. En primer lugar, pensamos en hacer uso de una base de datos, pero nos pareció innecesario ya que es muy poca información. Por tanto, decidimos usar la clase `SharedPreferences`. Esta clase almacena todos los tipos de datos primitivos, ya sea booleanos, enteros o cadenas de texto, entre otros, que se pretenden conservar de forma permanente. Estos datos se almacenan en forma *clave-valor*. Por ejemplo, si el usuario ha seleccionado la opción de *Definiciones* se guardará como clave el `String` "definiciones" y como valor `true`. En caso de que no se seleccione, se guardará con el valor `false`. Cabe destacar que en la herramienta no se almacenará ningún otro tipo de dato.

5.2.2. Servicios web externos

En el capítulo 3 explicamos qué son los servicios web (ver apartado 3.2). En *LeeFácil* accedemos a cuatro servicios web REST que están implementados por terceras personas. Esto ha facilitado el desarrollo de la aplicación, ya que hemos podido utilizar algunas funcionalidades sin necesidad de implementarlas desde cero.

Para realizar las peticiones a los servicios web se han implementado cuatro clases que podemos encontrar en el paquete `conexionServidor` y son: `ConexionGrafeno`, para establecer la conexión con Grafeno; `ConexionSpacy`, para API PLN NIL; `ConexionPICTAR`, para PICTAR y `ConexionAPI`, para API Accesibilidad. Debemos mencionar que las URLs de los servicios a los que se acceden se han almacenado en ficheros de textos. Esto se ha hecho así con la intención de facilitar al programador la modificación de la URL, en caso de que cambie, sin necesidad de tocar el código. Todos estos ficheros se encuentran dentro de la carpeta `raw` dentro del directorio `res`. El nombre del fichero corresponde con el nombre del servicio. A continuación, explicaremos cada uno de los servicios web utilizados en el presente trabajo.

5.2.2.1. Grafeno

Como se comentó en apartados anteriores, se ha implementado la opción de hacer un resumen del texto capturado, y para ello hacemos uso de Grafeno. Grafeno[1] es una librería Python creada por Antonio F. García Sevilla y Alberto Díaz Esteban. Puede hacer construcciones de grafos semánticos a partir de árboles de análisis sintáctico que son generados por tecnologías como SpaCy. Además, tiene la capacidad de hacer este proceso a la inversa y generar sentencias en lenguaje natural a partir de grafos semánticos.

[1] Para más información acceder a `https://github.com/agarsev/grafeno`

Actualmente, Grafeno sigue siendo un proyecto en progreso, pero se ha demostrado que puede ser muy útil para funcionalidades como las de resumir un texto. En nuestra aplicación no usamos esta librería directamente, sino que accedemos a un servicio web REST localizado en:

```
https://sesat.fdi.ucm.es/grafeno/run/summary_es
```

El servicio Grafeno es llamado desde la vista que aparece en la Figura 5.2c. Para poder acceder a este servicio hay que establecer una conexión POST en la que se le pasará como parámetro el texto original capturado. Por tanto, dentro de la clase `ConexionGrafeno` se creará un JSON con un campo `text` que contendrá el texto que se pretende resumir. Después de haber establecido una conexión se devolverá un `String` en formato JSON con un campo `result` que tendrá el resumen del texto.

Por último, se procesa el resultado de la operación, en este caso el resumen del texto se almacenará en una variable tipo `String`.

5.2.2.2. API accesibilidad

Para poder implementar la funcionalidad de la opción *Palabra* dentro de nuestra aplicación, hemos decidido hacer uso de la API Accesibilidad[2] creado por Plaza Estévez et al. (2016) como Trabajo de Fin de Grado. API Accesibilidad es una API de servicios que se crea con el objetivo de unificar varios servicios de accesibilidad en una misma aplicación y así facilitar el trabajo a los desarrolladores. Este servicio podrá devolver, como resultado, tanto JSON como XML. El resultado de la solicitud dependerá del formato que se introduzca en la url. API Accesibilidad cuenta con siete tipos de funcionalidades, pero en el presente trabajo solo se han utilizado las siguientes:

- *Sinónimos*: devuelve todos los sinónimos de la palabra introducida en la url. En caso de que esta no tenga sinónimos, devolverá un campo vacío. Para acceder se usará la siguiente url:

  ```
  https://sesat.fdi.ucm.es/tfgapi/servicios/rest/sinonimos
  /json/‘‘palabra_seleccionada’’
  ```

  En la url se cambiará `palabra_seleccionada` por el parámetro (en este caso una palabra) que se quiera analizar. Este procedimiento será igual para todos los servicios de API Accesibilidad. De la misma forma, si queremos que el resultado sea en XML, se escribirá `xml` en lugar de `json`.

[2] Para más información acceder a `http://sesat.fdi.ucm.es:8080/Web/`

- *Antónimos*: de la misma forma que el servicio anterior, dada una palabra devolverá todos los posibles antónimos de la misma. En caso de que no haya antónimos, devolverá un campo vacío. Podemos acceder a través de la siguiente url:

  ```
  https://sesat.fdi.ucm.es/tfgapi/servicios/rest/antonimos
  /json/‘‘palabra_seleccionada’’
  ```

- *Definición*: este servicio devolverá todas las acepciones de una palabra. Podemos acceder al mismo mediante la siguiente url:

  ```
  https://sesat.fdi.ucm.es/tfgapi/servicios/rest/definicion
  /json/‘‘palabra_seleccionada’’
  ```

Desde las vistas de la Figura 5.4 se puede acceder a API Accesibilidad siempre y cuando se tengan activadas las opciones *Definición*, *Sinónimos* o *Antónimos*. Cuando se invoque la clase `ConexionAPI` se le pasará como parámetro la palabra que se quiera analizar y el tipo de funcionalidad. A partir de estos datos se construirá la URL y se establecerá la conexión con el servicio. El valor devuelto será un `String` en formato JSON que será procesado.

5.2.2.3. API PLN NIL

En el capítulo 3 hablamos sobre el Procesamiento de Lenguaje Natural. Para desarrollar nuestra aplicación hemos tenido que usar tareas de procesamiento de lenguaje natural como la lematización de las palabras, ya que para usar API Accesibilidad necesitamos el lema de las mismas. Por ejemplo, si el usuario no entiende la palabra *perritos* que aparece dentro del texto capturado y quiere buscar su significado, no podemos pasar el parámetro *perritos* en la llamada al servicio de *Definiciones*. Por tanto, tendríamos que hallar el lema de la palabra, en este caso *perro*.

Sin embargo, no usamos directamente la librería SpaCy (ver apartado 3.4.2.2), sino que usamos un servicio web REST API PLN NIL[3] proporcionado por Antonio F. García Sevilla. API PLN NIL cuenta con tres servicios, de los cuales solo nos interesan los siguientes:

- Análisis morfológico de palabras: dada una palabra, devuelve el resultado de los análisis lingüísticos solicitados. De los análisis que hay disponibles usamos: análisis morfológico, el cual analiza la palabra y

[3] Podemos hallar toda la documentación en `https://nilgroup.github.io/nlp-api/`

devuelve su género, lema, numero, etc. y sinónimos, que devuelve todos los posibles sinónimos de la palabra. Ejemplo de cómo podemos acceder a este servicio:

```
https://holstein.fdi.ucm.es/nlp-api/analisis/‘‘palabra
?analisis1&analisis2...’’
```

Tenemos que cambiar `palabra` por la palabra que queremos analizar, y `analisis` por el tipo de análisis: `morfologico` o `sinonimos`. Se pueden solicitar varios análisis en una misma llamada, cada uno debe ir separado por un *ampersand* (`&`). Si no ponemos ningún análisis en la url por defecto devolverá el análisis morfológico. Por ejemplo, si queremos buscar todos los posibles sinónimos de la palabra *paz*, debemos escribir:
`https://holstein.fdi.ucm.es/nlp-api/analisis/paz?sinonimos`

- Análisis gramatical del texto: para acceder debemos crear un JSON con un campo `texto`, en el que incluyamos el texto a analizar. Al igual que para las palabras, se pueden solicitar varios tipos de análisis para la frase. Estos pueden ser:

  - `oraciones`: dado un texto devuelve una lista con cada una de las frases del texto.
  - `entidades`: tarea de reconocimiento de entidades nombradas.
  - `sintagmas`: tarea *chunking*.

  Además, se pueden solicitar todos los análisis para las palabras. En este caso se analizarán casa una de las palabras del texto.

```
https://holstein.fdi.ucm.es/nlp-api/analisis/‘‘?analisis1
?analisis2...’’
```

Cuando se invoca la clase `ConexionSpacy` se le pasan tres parámetros: texto o palabra a analizar, tipo de funcionalidad y tipo de análisis. API PLN NIL se llamará desde dos vistas distintas: desde la vista de la Figura 5.3a (*Palabra*) y desde la vista de la Figura 5.5a (*Frases*).

En cuanto a *Palabra*, accedemos a API PLN NIL pasándole `texto` como tipo de funcionalidad y `morfologico` como tipo de análisis. Haciendo la llamada con estos dos parámetros más el texto a analizar, el servicio devolverá un `String` en formato JSON con el análisis morfológico de cada una de las palabras del texto.

De forma análoga, en cuanto a *Frases*, se le pasará como parámetro `texto` como tipo de funcionalidad y `oraciones` como tipo de análisis. En este ca-

so, API PLN NIL dará como resultado una lista con las frases del texto analizado.

5.2.2.4. PICTAR

En capítulos anteriores hablamos, brevemente, sobre PICTAR como página web (ver apartado 2.2.3.2) y como servicio web de traducción (ver apartado 3.2.3.1). En este apartado nos centraremos un poco más en la segunda parte. En nuestra aplicación hemos implementado dos opciones: una en la que se puede traducir una palabra a pictograma, y la otra en la que se pueden seleccionar una o varias frases para traducirlas a pictogramas. Para hacer esta traducción utilizamos el servicio web de PICTAR. Para ello accedemos a:

```
https://sesat.fdi.ucm.es/serviciopictar/‘‘frase_o_palabra’’
```

cambiando `frase_o_palabra` por la palabra o las frases que se pretendan traducir.

Desde la vista de la Figura 5.5b y la vista de la Figura 5.4c accedemos al servicio web PICTAR. Para invocar la clase `ConexionPICTAR` solo se debe pasar como parámetro una cadena de texto, es decir, una palabra o conjunto de frases. Por ejemplo, si accedemos a `https://sesat.fdi.ucm.es/servicio pictar/La casa es grande`. Una vez llamado al servicio dará como resultado un `String` en formato de *array* de cuatro elementos. Cada elemento del *array* seguirá la siguiente estructura:

- Número de n-gramas: un número entero que representa la cantidad de n-gramas del elemento. Un n-grama es una subsecuencia de n elementos de una secuencia dada. Si aplicamos esto a textos podemos decir que son n palabras consecutivas, dentro de una frase, que pueden tener un significado. Retomando el ejemplo del apartado 3.2.3.1 vemos que la frase “saltar a la pata coja” es un 5-grama. En la base de datos de ARASAAC existen pictogramas que están asociados a una expresión completa. Si no se realiza este proceso de separación de n-gramas, se haría una traducción palabra a palabra y podría perderse el significado de la frase.

- Categoría gramatical: se distingue la palabra según su categoría gramatical: un determinante (`det`), un sustantivo o nombre (`noun`), un adjetivo (`adj`), un verbo (`verb`), un auxiliar (`aux`), un pronombre (`propn`), un participio (`part`), una interjección (`intj`), un adverbio (`adv`) o una conjunción (`conj`), entre otras.

- Identificadores de pictogramas: dado una palabra o una secuencia de

n-gramas pueden aparecer uno o varios identificadores de pictogramas. Estos identificadores vienen separados por comas y entre corchetes.

- Palabra o sucesiones de palabras.

Por tanto, después de realizar la petición con la frase "La casa es grande" el sistema devolverá:

```
[
    "0 det[7029, 8476] la",
    "0 noun[2317, 6964] casa",
    "0 aux[5581, 5858] es",
    "0 adj[4658] grande"
]
```

En nuestra aplicación, de cada uno de los elementos del *array*, solo nos interesaría el subelemento que tiene los identificadores de los pictogramas. Por tanto, cuando obtenemos el resultado anterior recorremos el *array* y guardamos los identificadores. Una vez que tenemos estos identificadores accedemos al servicio web que nos devolverá la imagen con el pictograma correspondiente:

```
http://hypatia.fdi.ucm.es/conversor/Pictos/‘‘identificador_del
_pictograma’’
```

Para mostrar los pictogramas, sin la necesidad de descargar las fotos en nuestro dispositivo, hacemos uso de Picasso. Picasso[4] es una librería que se encarga de la descarga y almacenamiento de imágenes en la caché del dispositivo. Está desarrollada por Square Inc. bajo la licencia Apache 2.0, y es muy conocida ya que solo requiere una línea de código para cada una de sus funciones. Para cargar una imagen, solo tenemos que llamar al método `load()` y pasarle la URL de la imagen.

5.2.3. Google Vision API

Una de las características que define nuestra aplicación es la captación de texto mediante el uso de la cámara del dispositivo móvil. Con tan solo enfocar el texto que queremos analizar, el móvil es capaz de reconocerlo y tras ser seleccionado convierte esa imagen a texto plano.

Para poder llegar a realizar esta transformación de imagen a texto plano nos hemos valido del uso de la tecnología OCR, de la cual ya hemos habla-

[4] Para más información acceder a `http://square.github.io/picasso/`

do en el capítulo de tecnologías. Entre las distintas implementaciones que ofrecían librerías para el uso de OCR optamos por usar Google Vision API.

Esta solución se trata de una serie de librerías proporcionadas por Google, las cuales nos permiten tener acceso a los métodos que convierte la cámara de nuestro teléfono móvil en analizador de texto en vivo. Una vez ha reconocido que lo que tiene delante el reconocedor de texto proporcionado por la API, será capaz de hacer una división del texto tal como se muestra en la 5.8:

- Bloques: conjuntos de líneas que forman un párrafo.
- Líneas: conjunto de palabras en el mismo eje.
- Palabras: conjunto de caracteres alfanuméricos.

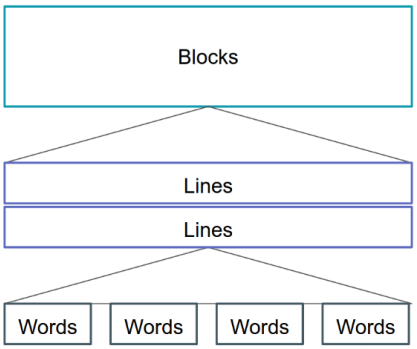


Figura 5.8: Esquema de la división del texto. Fuente: Mobile Vision

El dispositivo móvil reconocera el texto mediante la cámara según el esquema de bloques expuesto anteriormente, de tal manera que inicialmente empezara un reconocimiento por palabra y conforme mayor tiempo dejemos el dispositivo expuesto ante el texto este íra reconociendo las frases y parrafos. En la Figura 5.9 encontramos un ejemplo de como sería el proceso.

Una vez hemos seleccionado el texto a procesar, lo que haremos será guardar las coordenadas del eje X y eje Y del bloque de texto seleccionado. Una vez tengamos las coordenadas, llamaremos al método *mGraphicOverlay.getGraphicAtLocation(rawX, rawY)* lo que hacemos con este método es recuperar el bloque de texto y guardarlo en un objeto de tipo OCRGraphic, esto nos permitirá convertirlo en un elemento legible por Android y permitirnos convertir la imagen a texto.

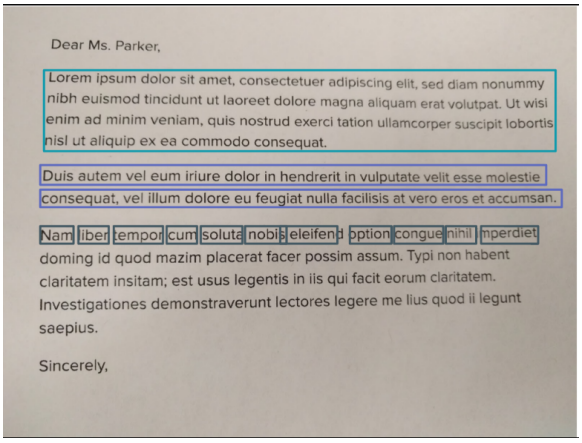


Figura 5.9: Esquema de la división del texto. Fuente: Mobile Vision

Capítulo 6

Evaluación con los usuarios

"No podemos resolver problemas pensando
de la misma manera que cuando
los creamos"
— Albert Einstein

En este capítulo se comentará, detalladamente, cómo ha sido el proceso de evaluación de la aplicación con usuarios reales. De igual manera se explicarán los resultados que se obtuvieron a partir de la misma.

6.1. Objetivos

A principios de mayo nuestra aplicación ya estaba terminada y por tanto, pudimos hacer una evaluación con usuarios y expertos reales. Con esta evaluación se pretendía conocer cuáles eran las opiniones que tenían los usuarios sobre la herramienta y, además, si cumplía o no con el objetivo de usabilidad y utilidad. Los principales aspectos a considerar en la evaluación fueron:

- Uso: analizar cuáles son los principales usos que le darían los usuarios. Comprobar, además, si sólo se centran en alguna o algunas funcionalidades en específico o si usarían todas las funcionalidades que ofrece la misma.
- Contexto: comprobar en qué situaciones y durante cuánto tiempo los usuarios usarían la herramienta.
- Intuitividad: analizar si nuestra aplicación presenta un diseño cómodo y natural.
- Carencias: evaluar si el usuario echa en falta alguna funcionalidad, otros botones o algún atajo, etc.

- Utilidad o beneficios: analizar si los usuarios que realizan la evaluación utilizarían esta aplicación en otras situaciones. Además, comprobar si la herramienta proporcionaría algún beneficio en su día a día.

6.2. Descripción

Para llevar a cabo la evaluación volvimos al Colegio Estudio3 AFANIAS. Esta vez nos recibió uno de los expertos (un profesor) que estuvo cuando hicimos la primera reunión. Los usuarios finales con los que se iba a probar la herramienta fueron nueve alumnos del colegio. Estos alumnos tenían entre 16 y 18 años y todos presentaban distintas discapacidades intelectuales. Dado que nuestra aplicación está pensada para que el usuario pueda interpretar cualquier texto que se encuentre en cualquier ámbito, fuimos a probarla en los alrededores del colegio.

Para ello el profesor preparó una especie de juego, tipo *Yincana*[1], en la que los alumnos tenían que leer unos carteles y cumplir con las indicaciones de los mismos. Esta era una de las actividades que se podían realizar para probar la aplicación, ya que algunos de los alumnos no saben o tienen dificultades para leer. Por tanto, podrían hacer uso de las funciones de lectura en voz alta o traducción a pictogramas que ofrece la herramienta.

Al empezar la evaluación, el primer alumno que usó la aplicación tuvo dificultades a la hora de capturar el texto. Como se comentó en el capítulo 6, cuando queremos capturar un texto, debemos enfocar nuestro móvil al mismo y en la pantalla aparecerá un recuadro blanco con el texto señalado. Este proceso puede tardar un par de segundos hasta que el sistema reconozca bien todas las letras o palabras. Tuvimos que explicar a los usuarios cómo se hacía e incluso ayudar a algunos a seleccionar el texto ya que este proceso no parece ser demasiado intuitivo para ellos. Sin embargo, según el experto, no cree que haya dificultad en que aprendan a capturar el texto de esa manera, pero para conseguirlo tendrá que haber un entrenamiento previo por parte de los usuarios. Otro de los problemas encontrados fue que los carteles tenían mucha separación entre una frase y otra y, por tanto, el sistema no capturaba el texto completo, sino que se creaba un recuadro para cada frase. Esto suponía que, en ocasiones, los alumnos tenían que capturar el texto frase a frase e ir analizándolas.

De las funciones que ofrece la aplicación las que más se utilizaron fueron lectura en voz alta y traducción a pictogramas. La mayoría de los usuarios que usaron la función de lectura en voz alta entendieron lo que el sistema

[1] Adaptación gráfica de la voz anglo-hindú *gymkhana*, conjunto de pruebas de destreza o ingenio que se realiza por equipos a lo largo de un recorrido, normalmente al aire libre y con finalidad lúdica.

leía y, además, cumplían con las indicaciones de los carteles. Sin embargo, otros alumnos tenían más dificultades para comprender la lectura e hicieron uso de la función traducción de frases a pictogramas. Uno de los principales problemas por el cual no entendían el texto hablado era debido a que la lectura en voz alta se hacía demasiado rápida. El resto de las opciones que ofrece la aplicación no pudieron ser usadas ya que este grupo de alumnos presentaban dificultades para leer. El experto comentó que, en un futuro, podría hacerse una evaluación con grupos de alumnos lectoescritores y de esta forma saber si son útiles funciones como las de Resumen, Definiciones, Sinónimos, Antónimos, entre otras.

6.3. Cuestionarios

Para completar la evaluación se hicieron dos modelos de cuestionarios, uno para los usuarios y otro para los expertos. El objetivo de los mismos era conocer la impresión de todas las personas que habían probado la aplicación. De esta forma podríamos ver si realmente se conseguía su objetivo.

Con respecto al cuestionario de los usuarios, las preguntas que se hicieron fueron las siguientes:

- ¿Te ha ayudado la aplicación en la actividad?
- ¿Te gustaría poder usar siempre la aplicación en tu móvil?
- ¿Has usado la opción de Escuchar texto? ¿Te ha ayudado?
- ¿Has usado la opción de Mayúsculas? ¿Te ha ayudado?
- ¿Has usado la opción de Traducción de frases a pictogramas? ¿Te ha ayudado?
- ¿Has usado la opción de Resumen? ¿Te ha ayudado la opción de Resumen?
- ¿Has usado la opción de Definiciones? ¿Te ha ayudado la opción de Definiciones?
- ¿Has usado la opción de Sinónimos? ¿Te ha ayudado la opción de Sinónimos?
- ¿Has usado la opción de Antónimos? ¿Te ha ayudado la opción de Antónimos?
- ¿Has usado la opción de Traducir palabras a pictogramas? ¿Te ha ayudado la opción de Traducir palabras a pictogramas?

- Algo más que nos quieras decir....

Para responder a estas preguntas se usó una especie de respuesta en semáforo que incluía emoticonos con caritas como la que podemos ver en la Figura 6.1. En esta respuesta el verde con el emoticono feliz representa mucho; el amarillo con el emoticono indiferente, regular; y el rojo con el emoticono triste, poco.

Figura 6.1: Respuesta en semáforo

Con respecto al cuestionario de los expertos el formato de respuesta era libre. Las preguntas que se incluyeron fueron:

- ¿Crees que les ha ayudado la aplicación en la actividad?
- ¿Crees que usarían la aplicación en su día a día?
- ¿Te gustaría disponer de la aplicación para otras actividades?
- ¿En qué otros escenarios podrían usar la aplicación para que les ayudara?
- ¿Qué cambiarías? ¿Falta algo? ¿Sobra algo?
- Valoras las opciones que les ofrece la aplicación:
  - Escuchar el texto en voz alta
  - Pasar el texto a Mayúsculas/Minúsculas
  - Traducción de frase a pictogramas
  - Resumen
  - Definiciones
  - Sinónimos
  - Antónimos
  - Traducción de palabras a pictogramas
- ¿Echas en falta alguna opción?
- Comentarios adicionales

Los cuesionarios se rellenaron en papel pero se han escaneado para facilitar su análisis. Para verlos podemos acceder al siguiente enlace: `https://drive.google.com/open?id=1HNLuEJNhjcVDM6Fj8ssNagrZC5pwhwFR`

6.4. Resultados

Después de haber realizado la evaluación y analizar las encuestas podemos llegar a la conclusión de que, pese a algunas limitaciones que nos fuimos encontrando durante toda la evaluación, la herramienta cumplía con su objetivo en cuanto a usabilidad y utilidad. En los cuestionarios vimos que todos los alumnos habían marcado el color verde como respuesta a si les había ayudado las funciones que habían usado en la aplicación. Cabe destacar que durante la evaluación pudimos observar lo ilusionados que estaban cuando nos daban las gracias y comentaban lo mucho que les gustaba la aplicación. Algunos de ellos incluso expresaron su deseo de utilizarla fuera del ámbito escolar.

Por otra parte, el experto presente en la evaluación comentó que la herramienta sería mejor utilizarla fuera del centro, ya que en los colegios suele haber restricciones respecto al uso del teléfono móvil. Por lo tanto, comentó que resultaría interesante usar la aplicación para realizar otros tipos de actividades como excursiones. Afirmó que el uso de la aplicación podría ayudarles en el desarrollo de su autonomía, ya que con ella podían afrontar cualquier texto que se encontrasen en su vida cotidiana sin la necesidad de tener que pedir ayuda a alguien. Además, nos recomendó añadir otras opciones para que la herramienta se adapte mejor a los usuarios. Algunas de estas son:

- Configurar la opción de lectura en voz alta para que leyera el texto más despacio.
- Añadir pictogramas a los botones. De esa forma los usuarios que tienen dificultades lectoras podrían entender mejor las funcionalidades de la herramienta.
- En la traducción a pictogramas, añadir la opción de lectura en voz alta. Así el usuario podrá escuchar el texto y ver su traducción a pictos a la vez.
- Hacer más grandes las imágenes de los pictogramas. En los dispositivos que tengan una pantalla pequeña no se puede visualizar bien el pictograma, por tanto sería interesante poder añadir una opción de ampliar el pictograma o hacerlos un poco más grandes.
- Hacer que la herramienta pueda reconocer letra cursiva.

Por todo lo anterior comentado consideramos que la evaluación con usuarios y expertos ha aportado un gran valor a nuestro trabajo. A nivel personal nos ha parecido una experiencia muy enriquecedora, ya que hemos podido comprobar que la herramienta que hemos desarrollado ayudará a mejorar la calidad de vida de muchas personas.

Capítulo 7

Conclusiones y Trabajo Futuro

"Nunca se es demasiado viejo para establecer
un nuevo objetivo, o para soñar
un nuevo sueño"
— CS Lewis

Poniendo punto y final al desarrollo del proyecto llegamos a una serie de conclusiones e ideas que se podrían tener en cuenta para futuros desarrollos.

7.1. Conclusiones

Con la idea base de querer crear una aplicación con la que poder ayudar a aquellas personas que sufren alguna discapacidad cognitiva en la compresión lectora, tuvimos que aprender como afecta dicha discapacidad a las personas que las sufren. Para este periodo de aprendizaje inicial contamos tanto con la ayuda de nuestras tutoras Raquel y Susana, así como con un grupo de expertos del Colegio Estudio3 AFANIAS.

Una vez acabado el período de aprendizaje habíamos aprendido que las discapacidades cognitivas para la compresión lectora, son una clase de trastorno que no tienen porque verse reflejado en el mismo grado en todas las personas que lo sufren. Partiendo de este punto teníamos que valorar que la aplicación fuese versátil, es decir, no valía generalizar la aplicación, sino que esta debía de ser personalizable para cada persona que la usara, y teníamos que valorar qué funcionalidades eran las más relevantes.

Otro punto importante ha sido el desarrollo de la interfaz. Hemos llevado a cabo un diseño sencillo y creemos que intuitivo para el usuario, encontrándonos con una serie de pantallas donde se intenta mostrar la información de la forma más clara posible al usuario, facilitándole así el uso de aplicación.

Además como comentábamos antes la interfaz es personalizable. Pese a que inicialmente la aplicación ofrece unos servicios preestablecidos, el usuario podrá elegir cuales quiere añadir o quitar de la interfaz, según el uso que vaya a hacer de la aplicación. Todo esto se consigue siguiendo un diseño centrado en el usuario.

Por último, como queríamos que se tratase de una aplicación móvil, pues de este modo era más sencillo que la gente la pueda usar durante el día a día, optamos por Android un sistema operativo que esta presente en la mayoría de los smartphones que encontramos en el mercado. Hay que añadir que la aplicación se encuentra disponible en Google Play Store, para que toda aquella persona que tenga la necesidad o le pueda servir de ayuda en algún momento, la tenga a su alcance.

7.2. Trabajo futuro

Una vez hemos finalizado el proyecto, observamos algunas mejoras o puntos a tener en cuenta para futuros desarrollos de la aplicación:

- Extender la aplicación a otros idiomas. Actualmente la aplicación solo funciona en español, pero sería un punto a valorar traducirla a otros idiomas para fomentar su uso.
- Mejora del sistema OCR. Esta mejora iría orientada a la hora de que no solo capte textos escritos a ordenador, sino que fuese capaz de reconocer la escritura realizada a mano.
- Mejorar la interfaz del modo “Palabras”. Conseguir que cuando dividimos un texto por la distintas palabras que lo forman, estas queden distribuidas de tal manera que se mantenga una mejor compresión del mismo.
- Añadir pictogramas a los botones. De esa forma los usuarios que tienen dificultades lectoras podrían entender mejor las funcionalidades de la herramienta.
- En la traducción a pictogramas, añadir la opción de lectura en voz alta. Así el usuario podrá escuchar el texto y ver su traducción a pictogramas a la vez.
- Hacer más grandes las imágenes de la vista de Traducción a pictogramas.
- Que la herramienta pueda reconocer letra cursiva.

Chapter 7

Conclusions and Future Work

At the conclusion of this project, we have found several themes and ideas that could be considered for future development.

7.1. Conclusions

With the main idea of this project being to create a mobile application in order to help people who suffer from cognitive disabilities, we had to learn about the different effects this issue can produce. During this time, we had the help of our form teachers Raquel and Susana, as well as some specialists from Colegio Estudio3 AFANIAS.

Once this process was completed, we had learn that the cognitive disabilities to the reading comprehension are a type of disorder whose effects may vary from one person to another. For this reason, we thought about versatility of the application, that is to say, we have to develop a mobile application that users can customize and we had to evaluate which functionalities were the most relevant.

Another important point have been designing the interface. We have created a user-centered design that is user friendly. In the interface of our mobile application, we tried to show the information clearly to help the users. Even though the application have some predetermined functionalities, users can change these ones by others that could be more useful for them.

Finally, we wanted to implement a mobile application, considering that it is the best way to encourage daily use. To accomplish this objective, we use Android OS since it can be found in the majority of smartphones that people use in their daily lives. Furthermore, the app is available in the Google Play Store, to further increase accessibility.

7.2. Future work

Once the project was finished, we found some points or improvements that could be interesting to further develop and improve including but not limited to:

- More languages added to the application. Nowadays the mobile application only works in Spanish. It could be a great idea to translate it to others languages to support that people use it.
- Improve the OCR system. This improvement will focus on improving the camera to recognize hand-printed texts.
- Improve the interfaces in the "Words" option. Try to get a better distribution of the division of the text in words. The principal objective of this improvement is keep the context.
- Include pictograms in the buttons. It would make easier for some users to recognize the functionalities of the mobile application.
- Include a translation for voice reader to pictograms. This options would be beneficial for the users, considering that they would be able to listen to the text and see the pictograms at the same time.
- Extend the available pictures that can be used for translation into pictograms.
- Recognize italics texts.

Capítulo 8

Contribuciones al proyecto

8.1. Elianni Agüero Selva

Para poder hablar de mi contribución al proyecto es importante hacer una diferenciación entre la fase de investigación y la parte de implementación de la aplicación.

En primer lugar, hicimos un reparto de tareas en el que yo comencé buscando información acerca de la discapacidad cognitiva y cómo es el día a día de las personas que sufren este tipo de trastorno. Indagando pude hacerme una idea aproximada de los problemas y limitaciones que tienen a la hora de comunicarse con los demás e interactuar con su entorno. Afortunadamente, existen los Sistemas Alternativos y Aumentativos de Comunicación (SAAC) cuyo objetivo es intentar ayudar a eliminar estas barreras en la comunicación. De estos SAACs me centré en estudiar los sistemas pictográficos que existen actualmente para poder decidir cuáles eran los más indicados para utilizar en la aplicación. Como alternativa a los SAAC, investigué acerca de la Lectura Fácil y qué pautas se deberían seguir para adaptar un texto. Esto está relacionado con la opción de *Resumen* de la aplicación. Todo esta fase de investigación me sirvió para realizar el tema del estado del arte de la memoria.

Con respecto a la implementación, se decidió desarrollar la aplicación en Android Studio. A pesar de conocer el lenguaje de programación Java y de haber tratado con XML en algunas asignaturas de la carrera, me supuso un gran esfuerzo aprender a utilizar este IDE.

Después de haber hecho la fase de investigación me puse con la implementación de la aplicación. Como base sobre la que empezar, se utilizó un proyecto que accedía a la cámara del dispositivo, podía reconocer texto y al pulsar sobre él se guardaba en una variable de tipo `String`.

A su vez, me puse a investigar cómo poder acceder a los servicios que queríamos utilizar en la aplicación, ya que este proceso era completamente desconocido para mí. El proceso de conectar con los servicios web no fue una tarea trivial, ya que estuve varios días intentando establecer la conexión sin ningún resultado. Por un lado, por cómo está configurado Android tenía que crear un hilo secundario para poder establecer la conexión, ya que no dejaba hacerlo en el hilo principal como yo lo estaba intentando. Por otro lado, al intentar establecer una conexión HTTPS, daba un error de certificados de la conexión. Esto era debido a que el servidor no estaba bien configurado y faltaba un certificado intermedio que los navegadores no necesitan, pero Android sí. Esto lo solucioné añadiendo dicho certificado dentro de la aplicación como un certificado de confianza. También hubo que configurar el uso de conexiones HTTP ya que a partir de la versión 9.0 de Android hay que establecer que se van a utilizar conexiones no cifradas.

De forma paralela, empecé a redactar el tema 3 de la memoria comentando qué tecnologías estábamos usando en el proyecto y diseñé varios prototipos de interfaz para la aplicación. Fueron varias semanas las que estuvimos modificando los prototipos ya que se fueron incorporando nuevas funcionalidades al proyecto. Después de varias iteraciones y modificaciones de los mockups solo quedaron dos prototipos finales, uno por cada integrante de este trabajo.

Para validar el diseño de la aplicación, se concertó una reunión con los profesores del Colegio Estudio3 AFANIAS en la cual expliqué el objetivo de la aplicación y mostré las propuestas que teníamos. En dicha reunión los expertos comentaron algunos aspectos a modificar en la interfaz para que esta sea más accesible a usuarios finales.

Una vez hechos los cambios en la interfaz y de haber redactado el apartado de evaluación con los expertos en la memoria, me puse a aprender a diseñar las vistas en Android Studio. Me fui guiando por el prototipo final que teníamos para desarrollar la interfaz, aunque hubo algunos aspectos, como el tamaño de botones o la posición en la que estaban, que fui modificando ya que no quedaban bien o no parecían demasiado intuitivos. Fueron varias semanas realizando la interfaz y conectándola con los servicios que nos habían proporcionado, hasta que logré terminarlo.

En este punto solo quedaba implementar la opción de configuración de funcionalidades. Mientras tanto me puse a redactar la primera parte del tema de implementación en la memoria, concretamente la descripción de la aplicación y los dos primeros apartados de la arquitectura.

A principios de mayo se había concertado otra cita con el Colegio Estudio3 AFANIAS para realizar la evaluación con usuarios reales. Finalmente, como la configuración de funcionalidades no funcionaba correctamente, investigué y utilicé `SharedPreferences`. Llegado el día de la evaluación la aplicación estaba lista y publicada en *Google Play Store*. Con respecto a es-

to, realicé todo el proceso necesario para la publicación. Antes que nada, hubo que abrir una cuenta en Google como desarrollador y para ello se usó un correo del grupo de investigación NIL. Después tuve que generar el fichero *.apk* y para ello había que crear un certificado de confianza. Para la publicación de la aplicación se pedía además un logo para la aplicación, capturas de pantallas, añadir descripciones, etc.

Participé en la evaluación como espectadora tomando notas de todo lo que veía en ella: si lo usuarios se desenvolvían bien con la aplicación, si tenían problemas para usarla, si les había ayudado, etc. Al final de la misma recogí *feedback* tanto de los usuarios como de los expertos. Tras la evaluación redacté el tema 6 de evaluación con los usuarios.

8.2. Ignacio Sande Soltero

Como el proyecto se iba a basar en el desarrollo de una aplicación móvil y el entorno operativo que habíamos elegido era Android, debido a que queríamos llegar a un gran número de usuarios, tuvimos que aprender a programar sobre él. Este aprendizaje no resulto muy difícil, ya que se basa principalmente en Java, donde ya tenía una base, y XML un lenguaje de marcas utilizado principalmente para el desarrollo de la interfaz gráfica de la aplicación.

Uno de los elementos de los que se alimenta nuestra aplicación es del uso de la cámara del dispositivo móvil que estemos usando para poder captar el texto que queremos analizar. Para conseguir que el dispositivo fuese capaz de reconocer cuando está captando un texto era necesario utilizar la tecnología OCR con la cual nunca había trabajado anteriormente.

Con respecto al mundo de OCR, con una pequeña búsqueda que realicemos, podemos encontrar diferentes soluciones que nos permitirán que la cámara reconozca que lo que está enfocando es un texto escrito. Tras valorar las distintas alternativas, utilice la API proporcionada por Google (Google Vision), la cual esta muy bien documentada con tutoriales y repositorios de ejemplo, los cuales sirvieron de apoyo para la correcta implementación de la captación del texto con nuestro dispositivo Android.

Para la gestión de datos persistentes por parte de la aplicación nos planteamos el uso de una pequeña base de datos, pero no queríamos que dependiera de un servicio externo, sino que se gestionara dentro del propio dispositivo móvil, de esta manera descubrí que Android cuenta con SQLite.

SQLite se trata de un pequeño motor de bases de datos, el cual nos permitía mantener de forma permanente determinados valores de la aplicación como podían ser los ajustes de personalización de la interfaz del usuario. Pese a tratarse de una biblioteca Java y que su implementación se basaba

en la creación de una clase con los distintos métodos necesarios para la visualización y modificación de los datos decidí descartarlo, ya que, tras varias pruebas no conseguimos que el funcionamiento se ajustara a lo deseado y optamos por otra vía de desarrollo que nos proporcionase la misma solución.

También nuestra aplicación cuenta con una serie de funcionalidades, las cuales se tratan de una serie de servicios REST. Debido a que era la primera vez que trabajába con este tipo de servicios y más aún desde Android, tuve que investigar como era la implementación para poder llamar desde la aplicación a un servicio externo mediante una URL de tipo HTTP o HTTPS. Creando una primera versión del código que nos permitiese la conexión.

Para la conexión con aquellos servicios que partíamos desde una URL de tipo HTTP, hubo alguna pequeña dificultad debido a que la llamada se debía hacer en un hilo paralelo al que corría la ejecución principal de la aplicación. Para los servicios HTTPS encontré mayores dificultades, ya que, la empresa certificadora no se trata de una empresa que esté reconocida por Google. Por tanto, para conectar a estos servicios no servía el mismo método que habíamos utilizado anteriormente, sino que tuvimos que descargarnos el certificado e incluirlo dentro de la aplicación, para que de esta manera Android lo tomara como un certificado de confianza y así poder conectarse.

Por otro lado, teníamos que pensar cual era el aspecto que queríamos darle a la aplicación y que este diseño se ajustase a las necesidades del usuario. No valía que la pantalla estuviese muy sobrecargada de información, o que no fuese intuitiva y que la persona que la estuviese utilizando se perdiese o que el acceso a las funcionalidades no fuese claro.

Para llegar a cumplir con este propósito tanto mi compañera como yo realizamos inicialmente unos primeros modelos mediante mockups, para la implementación de dicho modelo hice uso de la herramienta Balsamiq Mockups 3. Con el primer modelo realizado y tras reunirnos con nuestras tutoras del proyecto observamos cuales eran los puntos por mejorar, de tal manera que fui modificándolo hasta crear un modelo final de la aplicación, donde nos ajustábamos plenamente a un diseño sencillo e intuitivo, para que al usuario le fuese fácil usar la herramienta y la pueda terminar incorporando a su día a día.

Como aportación al desarrollo de la memoria me he encargado de la redacción de algunos de los puntos que se exponen, estos son:

- Introducción: donde vemos reflejadas las razones por las cuales decidimos optar por un proyecto de este tipo, cuales son los objetivos que teníamos y como hemos estructurado este documento.

- Diseño de la aplicación: en este punto hablamos sobre todo el proceso realizado para conseguir una interfaz sencilla y amigable con el usuario.

- Resumen: una breve sinopsis donde se refleja el trabajo realizado en el proyecto.
- Conclusiones y trabajo futuro que queda por implementar o posible mejoras que se pueden llevar a cabo.

Bibliografía

Y así, del mucho leer y del poco dormir,
se le secó el celebro de manera que vino
a perder el juicio.

Miguel de Cervantes Saavedra

Aluisio, S., Specia, L., Gasperin, C. y Scarton, C. Readability Assessment for Text Simplification. En *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, páginas 1–9. Association for Computational Linguistics, 2010.

Anula, A. y Belinchón, M. Don quijote de la mancha. edición de fácil lectura. Disponible en `http://www.lecturafacilextremadura.es/wp-content/uploads/bsk-pdf-manager/quijote_facil_lectura_1.pdf`.

ARASAAC: Gobierno de Aragón. Portal aragonés de la comunicación aumentativa y alternativa. s.f. Disponible en `http://www.arasaac.org/aac.php` (último acceso, 2018).

Asociación Americana de Psiquiatría. Manual diagnóstico y estadístico de las enfermedades mentales. *Arlington, VA: American Psychiatric Publishing*, 2013.

Baldassarri, S., Rubio, J. M., Azpiroz, M. G. y Cerezo, E. AraBoard: Multiplatform Alternative and Augmentative Communication Tool. En *Proceedings of the 5th International Conference on Software Development for Enhancing Accessibility and Fighting Info-exclusion, DSAI 2013, University of Vigo, Spain, November 13-15, 2013*, páginas 197–206. 2013.

BBVAOpen4U. API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos. 2016. Disponible en `https://bbvaopen4u.com` (último acceso, 2019).

Belloch, C. Las Tics en Logopedia: Audición y Lenguaje. s.f. Disponible en `https://www.uv.es/bellochc/logopedia/NRTLogo1.wiki` (último acceso, 2018).

Booth, D., Haas, H., McCabe, F., Newcomer, E., Michael, I., Ferris, C. y Orchard, D. Web services architecture. 2004. Disponible en `https://www.w3.org/TR/ws-arch/` (último acceso, 2019).

Bustos, A. ¿Qué es Pictograma? *Blog de Lengua*, 2010. Disponible en `https://blog.lengua-e.com/2010/que-es-un-pictograma/` (último acceso, 2018).

Cabello, F. y Bertola, E. Características formales y transparencia de los símbolos pictográficos de ARASAAC. *Revista de Investigación en Logopedia*, vol. 5(1), 2015. ISSN 2174-5218.

Callaham, J. The history of Android OS: its name, origin and more. *Android Authority*, 2018.

IIMED. ¿qué es el traductor google translate y para que sirve? s.f. Disponible en `https://iiemd.com/blog/es-traductor-de-google/que_es_google_translate_traductor_de_google` (último acceso, 2019).

Jiménez Corta, L. Herramienta de apoyo a la navegación web para personas con discapacidad, 2018. Universidad Complutense, Facultad de Informática, curso 2017/2018.

Juan, M. L., De Ramón, A., Ballesteros, E. y Romero, B. Dictapicto, rompiendo las barreras para la comunicación y la participación. *Tecnología accesible e inclusiva: logros, resistencias y desafíos*, Disponible en `http://diversidad.murciaeduca.es/publicaciones/tecno2017/doc/t04.pdf`.

Lloréns Macián, B. Introducción y enseñanza del sistema minspeak de comunicación aumentativa. guía práctica para el profesional. 2006.

Martín Guerrero, A. PICTAR: una herramienta de elaboración de contenido para personas con TEA basada en la traducción de texto a pictogramas, 2018. Máster en Ingeniería Informática, Facultad de Informática, Departamento de Ingeniería del Software e Inteligencia Artificial, curso 2017-2018.

NeoTeo. IBM Simon, el primer smartphone de la historia. *ABC*, 2012. Disponible en `https://www.abc.es/20120222/tecnologia/abci-simon-primer-smartphone-historia-201202221308.html` (último acceso, 2019).

Nilanchala. Android Studio Features. 2013. Disponible en `https://stacktips.com/tutorials/android/android-studio-features` (último acceso, 2019).

Plaza Estévez, S., Acosta Morales, C. y Ramírez Lamela, N. API de servicios web orientados a accesibilidad, 2016. Trabajo de Fin de Grado

en Ingeniería Informática (Universidad Complutense, Facultad de Informática, curso 2015/2016).

Ruiz Martín, C., Galván Calleja, P., Hervás Ballesteros, R. y Gervás Gómez-Navarro, P. *Editor predictivo de mensajes en pictogramas*. Universidad Complutense de Madrid, 2014.

Saggion, H., Gómez-Martínez, E., Etayo, E., Anula, A. y Bourg, L. Text simplification in Simplext: Making texts more accessible. *Procesamiento del lenguaje natural*, 2011.

Sandín Carral, G. A. OCR: Análisis e implementación de algoritmos en nuevas tecnologías de paralelización. *Trabajo de Fin de Grado*, 2015.

Tronbacke, B. et al. Directrices para materiales de lectura fácil. Disponible en `http://www.ifla.org/files/assets/hq/publications/professional-report/120-es.pdf`.

Universidad de Alicante. Introducción a los servicios web. invocación de servicios web soap. 2012. Disponible en `http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.pdf` (último acceso, 2019).

Universitat Politècnica de València. Arquitectura de android. s.f. Disponible en `http://www.androidcurso.com/index.php/recursos/31-unidad-1-vision-general-y-entorno-de-desarrollo/99-arquitectura-de-android` (último acceso, 2019).

Villena Román, J., Crespo García, R. M. y García Rueda, J. J. Procesamiento del Lenguaje Natural. 2011. Disponible en `http://ocw.uc3m.es/ingenieria-telematica/inteligencia-en-redes-de-comunicaciones/material-de-clase-1/09-procesamiento-del-lenguaje-natural` (último acceso, 2019).